



**Projet Industriel :  
Ecole Polytechnique Universitaire de Montpellier**

Informatique et Gestion : quatrième année d'ingéniorat  
Année académique 2011/2012



**Lancement du site scientific-profile.com  
Développement de la « viralité » web**

**RAPPORT TECHNIQUE**

**Réalisé par :**

Marylise Charlery  
Chen Yang Gao  
Joris Puechlong  
Julien Lessart

**Référents :**

Jacques Ruiz  
Clément Penin

## INDEX DU DOCUMENT

### **TITRE :**

Rapport Technique pour le lancement du site scientific-profile.com et du développement de la « viralité » web.

### **AUTEURS :**

Ce document a été rédigé par Chen Yang Gao, Joris Puechlong, Julien Lessart et Marylise Charlery étudiants en quatrième année d'ingéniorat Informatique et Gestion à l'Ecole Polytechnique Universitaire de Montpellier.

### **MOTS CLEFS :**

Expernova.com, scientific-profile.com, cartographie de compétences scientifiques, viralité, buzz, viadeo, linkedIn, réseaux sociaux, web communautaire, web 2.0, widgets, créativité, indépendance.

### **RESUME :**

Ce document est un rapport technique destiné aux professeurs de Polytech. Ce rapport contient les explications techniques et certains bouts de code de notre projet. Il va de pair avec le rapport de synthèse.

## TABLE DES MATIERES

1.	Introduction.....	4
2.	Première partie : connexion via Viadeo / LinkedIn .....	5
2.1.	Connexion via Viadeo .....	6
2.1.1.	Récupération des informations.....	6
2.1.2.	Extraction de l'id Viadeo de l'utilisateur .....	6
2.1.3.	Vérification de l'existence de l'utilisateur dans Scientific-Profile .....	7
2.1.4.	Enregistrement d'un nouvel utilisateur .....	7
2.2.	Connexion via LinkedIn.....	8
2.2.1.	Se connecter via LinkedIn API .....	8
2.2.2.	Récupération des informations de l'utilisateur sur linkedIn .....	9
2.3.	Vérification de l'existence de l'utilisateur dans Scientific-Profile .....	14
2.4.	Remplissage automatique du formulaire.....	15
2.4.1.	Récupérations des données .....	17
2.4.2.	Le Formulaire.....	18
3.	Implémentation des Shares.....	19
4.	Création de la page Facebook / Implémentation de la likebox .....	20
4.1.	Page Facebook.....	20
4.2.	Implémentation de la likebox de Facebook .....	20
5.	Création de la signature de mail. ....	21
5.1.	La signature .....	21
5.2.	Fonctionnement .....	23
5.3.	La signature en HTML.....	25
5.4.	La signature Texte .....	25
6.	Publications Mendeley .....	26
6.1.	Définition .....	26
6.2.	Choix et Technologies.....	26
6.3.	Connexion à l'API Mendeley .....	26
6.4.	Importation des documents.....	29

## 1. INTRODUCTION

Ce rapport va présenter la partie technique de notre projet industriel.

Notre projet a été orienté Web, avec à la clé l'implémentation de divers modules indépendants. Ainsi chaque partie de ce rapport présentera un module.

La mise en place des divers modules sur le site est à la charge de l'entreprise.

Pour tester nos divers bouts de code nous travaillons sous Windows et Linux, et nous avons installé de serveurs sur nos machines (WampServer, ZendServer Community Edition sous Windows et Apache sous Unix).

## 2. PREMIERE PARTIE : CONNEXION VIA VIADEO / LINKEDIN

Dans cette partie, nous allons expliquer comment nous avons réussi à implémenter la connexion au site Scientific-Profile grâce à la connexion via les réseaux sociaux professionnels que sont Viadeo et LinkedIn.

Le but de cette partie est de permettre à l'utilisateur d'effectuer le moins d'étape possible afin de se connecter au site Scientific-Profile ou de créer son compte. Ceci permettra de conquérir les utilisateurs qui abandonnent la création d'un compte parce qu'ils n'ont pas envie de perdre du temps à créer un énième compte sur un autre site. Nous avons choisis les réseaux sociaux Viadeo et LinkedIn parce que ce sont des réseaux professionnels et que nous avons besoins de données professionnelles semblables à celles que l'on peut trouver dans un CV comme les diplômes. Les données récupérées seront utilisées afin de remplir une grande partie des champs du profil utilisateur de Scientific-Profile.

Afin de créer un compte sur Scientific-Profile, nous avons besoin de récupérer certaines informations sur l'utilisateur qui sont les suivantes :

- L'email de l'utilisateur → L'id Viadeo/LinkedIn de l'utilisateur
- Le nom de l'utilisateur
- Le prénom de l'utilisateur
- Le lien vers la photo de profil de l'utilisateur
- Les études de l'utilisateur
- Les diplômes de l'utilisateur
- La présentation de l'utilisateur


Malheureusement, Les documentations des deux APIs stipulent qu'il est interdit de copier les données et de les enregistrer sans l'accord de l'utilisateur. C'est donc pour cela que nous avons créé un formulaire d'inscription pré-rempli avec les informations récupérées. L'utilisateur à le choix d'accepter ou non que l'on enregistre ses informations. De plus, il n'y a pas moyen de récupérer l'email de l'utilisateur pour des raisons de sécurité (anti spam). Nous avons donc choisi de remplacer l'email qui permettait d'identifier l'utilisateur par son id Viadeo/LinkedIn.

Ainsi, l'utilisateur aura le choix de se connecter sur le site soit par une connexion classique, soit via son réseau social favori comme on peut le voir sur cette image.



Le langage utilisé pour l'exploitation des API est le JavaScript car la documentation des API et les exemples sont présentés en JavaScript et leur utilisation grâce au JavaScript est très facilitée. De plus les Framework JavaScript de ses API sont facile à utiliser.

## 2.1. CONNEXION VIA VIADEO

Afin de nous connecter via le réseau social Viadeo grâce au bouton , nous avons travaillé avec l'API de Viadeo. Ainsi, nous avons utilisé le code JavaScript fourni sur le site [dev.viadeo.com](http://dev.viadeo.com) afin de créer le bouton de connexion.

Tout d'abord, nous avons besoin d'une clé-API (apiKey) que l'on peut obtenir grâce à un compte Viadeo. Une fois cette clé obtenue, nous avons pu commencer le travail.

Pour pouvoir utiliser les fonctionnalités JavaScript, il faut tout d'abord importer le Framework de l'API de Viadeo comme ceci :

```
<script src="http://cdn.viadeo.com/javascript/sdk.js"></script>
```

### 2.1.1. RECUPERATION DES INFORMATIONS

Ensuite, nous avons adapté le code exécuté lorsque l'utilisateur se connecte à son compte Viadeo afin de récupérer toutes les informations nécessaires. La fonction ci-dessous permet d'obtenir les informations :

```
VD.api('/me', {connections : 'education'}, function(r)
```

Le `"/me"` permet d'obtenir les informations sur l'utilisateur qui vient de se connecter sur Viadeo. L'option `"{connections : 'education'}"` permet d'obtenir les études de l'utilisateur en plus des autres informations que nous renvoie `"/me"`. Malheureusement l'API de Viadeo ne permet pas de récupérer les diplômes.

Les informations récupérées sont stockées dans un objet JSON, c'est la variable `"r"` dans l'exemple.

### 2.1.2. EXTRACTION DE L'ID VIADEO DE L'UTILISATEUR

Afin de vérifier si l'utilisateur est enregistré sur Scientific-Profile ou non, nous récupérons l'id de l'objet JSON récupéré précédemment grâce à la commande :

```
var id = r.id;
```

C'est cela qui nous permettra de différencier un utilisateur qui s'enregistre pour la première fois via Viadeo, d'un utilisateur déjà enregistré (dont on connaît l'id).

### 2.1.3. VERIFICATION DE L'EXISTENCE DE L'UTILISATEUR DANS SCIENTIFIC-PROFILE

Ensuite, nous appelons la page php "getid.php" qui va renvoyer un booléen avec la valeur Vrai si l'utilisateur existe dans Scientific-Profile (voir : 1.3. Vérification de l'existence de l'utilisateur dans Scientific-Profile).

### 2.1.4. ENREGISTREMENT D'UN NOUVEL UTILISATEUR

Dans le cas contraire, l'objet est transformé sous format texte et encodé. On envoie cet objet au formulaire d'inscription. (Voir : 1.4. Remplissage automatique du formulaire).

Voici la structure de l'objet récupéré contenant les informations de l'utilisateur. Pour des raisons de lisibilité, seules les informations nécessaires sont explicitées.


Légende :

- { } : Objet JSON
- [ ] : Tableau
- les données récupérées ne sont que des textes (string)
- **texte bleu** : contenu de la donnée

```
{
  "id", id viadeo
  "introduction", Présentation
  "educations":{
    "data":[
      {
        "school":{
          "name", Nom de l'école
          "department", Nom du département/section de l'école
        },
      },
    ]
  },
  "first_name", Prénom de l'utilisateur
  "last_name", Nom de l'utilisateur
  "picture_large", URL de l'image de l'utilisateur
}
```

## 2.2. CONNEXION VIA LINKEDIN

Afin de permettre la connexion via le réseau social LinkedIn, nous avons travaillé avec l'API de LinkedIn.

Nous devons utiliser le bouton officiel de connexion de LinkedIn  **Log in with LinkedIn**, à voir sur le site <http://developer.linkedin.com/apis>. Pour réaliser le projet, nous avons choisi d'utiliser l'API avec JavaScript fourni par LinkedIn. Dans la suite, nous vous présenterons les étapes nécessaires à la connexion via LinkedIn.

### 2.2.1. SE CONNECTER VIA LINKEDIN API

Afin d'utiliser les informations de l'utilisateur et de se connecter via LinkedIn, il faut passer par cinq étapes simples.

#### 1. Trouvez votre nom de domaine

La première étape, c'est de définir le nom de domaine. Dans notre cas, pour développer et pour tester, nous avons travaillé en local, nous avons donc utilisé "localhost".

#### 2. Obtenez une clé API

Pour obtenir la clé API, il faut visiter l'adresse suivante: <https://www.linkedin.com/secure/developer>.

Ensuite, il faut cliquer sur "Add New Application" et remplir les informations nécessaires, LinkedIn va générer une clé spécifique pour l'application créée. De toute façon, si vous voulez utiliser une clé API, il faut que vous vous inscriviez tout d'abord sur "LinkedIn Developer". (<http://developer.linkedin.com/>)

#### 3. Ajouter LinkedIn Framework à votre page

Dans la page, ajouter les codes suivants:

```
<script type="text/javascript" src="http://platform.linkedin.com/in.js">
  api_key: votre clé api
</script>
```

C'est la partie la plus importante pour faire fonctionner le Framework de LinkedIn API (JavaScript).

Pour faire cela, il faut vérifier si la fonction JavaScript dans le navigateur web est activée.

#### 4. Ajouter un bouton avec LinkedIn

Pour afficher le bouton  **Log in with LinkedIn**, ajoutez le code suivant:

```
<script type="IN/Login "></script>
```

C'est le bouton généré automatiquement par LinkedIn et c'est lui qui va faire apparaître une fenêtre qui demande accès au compte de l'utilisateur en lui demandant de rentrer son identifiant et son mot de passe. C'est un processus d'authentification sur OAuth2 standard, mais pour faire fonctionner l'API il n'est pas utile d'en connaître le détail.



## 5. Autorisation automatique

Pour éviter que la fenêtre d'authentification n'apparaisse à chaque chargement de page alors que l'utilisateur est déjà connecté sur son compte LinkedIn, il faut rajouter une ligne dans le framework importé " authorize: true".

Le code est:

```
<script type="text/javascript" src="http://platform.linkedin.com/in.js">
api_key: your_api_key_goes_here
authorize: true
</script>
```

### 2.2.2. RECUPERATION DES INFORMATIONS DE L'UTILISATEUR SUR LINKEDIN

#### 1. Utilisation de l'objet "IN" défini par LinkedIn

Pour accéder aux objets qui contiennent les informations de l'utilisateur qui s'est connecté, nous avons besoin d'utiliser l'API JavaScript fournie par LinkedIn. Elle est le pont entre le navigateur web de l'utilisateur et le terminal REST chez LinkedIn. Le fonctionnement de cet API consiste simplement à appeler une ressource et à retourner un objet JSON. L'objet contenant les informations de l'utilisateur est donc un objet JSON.

Tout d'abord, il faut changer le code pour importer le framework de l'API LinkedIn:


```
<script type="text/javascript" src="http://platform.linkedin.com/in.js">
api_key: DXjUrY-9apre6gnyK080MpTloIS4f38AleG08Y0kLM0DH2xeNQATIfDMuoisCMO_
onLoad: onLinkedInLoad
authorize: true
</script>
```

Remarquons que dans ce script, nous avons un nouveau paramètre : onLoad. Avec ce paramètre la fonction fait en sorte que lors du chargement de la page, le framework attende jusqu'à ce qu'il ait récupéré toutes les informations de l'utilisateur avant d'exécuter la suite des instructions.

On va ensuite regarder la fonction JavaScript "onLinkedInLoad"

```
function onLinkedInLoad() {
    IN.Event.on(IN, "auth", onLinkedInAuth);
}
```

L'objet IN permet de communiquer avec l'API JavaScript. Dans le code, la fonction principale est IN.Event, qui permet d'écouter les activités dans le système et de lancer la fonction onLinkedInAuth. Le paramètre "auth" indique que la fonction suivante, (définie par onLinkedInAuth) appartient à un processus d'authentification et qui est déclenché quand l'utilisateur se connecte.

Une remarque pour rappeler que pour effectuer ce processus d'authentification, il faut ajouter le bouton , en inscrivant `<script type="in/Login"></script>` dans votre code HTML.

## 2. Extraction des informations sur le profil d'un utilisateur

Après s'être authentifié, nous aurons le droit de retirer les informations d'un utilisateur. Il faut souligner que dans la fonction `onLinkedInLoad`, nous avons appelé une fonction "`onLinkedInAuth`", qui va être exécutée après le processus d'authentification.

Dans cette fonction :

```
function onLinkedInAuth() {  
    IN.API.Profile("me").result(displayProfiles);  
}
```

On a appelé une API `IN.API.Profile("me")`, qui va faire appel aux serveurs LinkedIn et récupérer les informations concernant le profil d'un utilisateur, ici "me". L'utilisateur "me" correspond à l'utilisateur qui est actuellement connecté à l'API LinkedIn. "result" est une fonction "callback" qui va premièrement exécuter la fonction `displayProfiles` puis stocker les informations qu'il a reçu de cette fonction.

## 3. Utilisation des "Profile Fields"

Afin de trouver et d'exploiter les informations définies par le demandeur Expernova parmi celles fournies par l'API, nous avons besoin de considérer d'avantages de paramètres. Ainsi nous avons trouvé ces données concernant l'utilisateur parmi les "Profile Fields".

Ce qu'on change:

Dans la ligne de commande `IN.API.Profile("me").fields("firstName", "lastName", "industry").result(displayProfiles)`, nous avons utilisé une fonction `field()` qui récupère les données des champs dont le nom est placé en paramètre. Ces champs ou "fields" en anglais, sont "firstName" pour le prénom, "lastName" pour le nom, et "industry" pour le secteur dans lequel l'utilisateur travaille. Pour consulter les options supplémentaires, voir le lien:

<http://developer.linkedin.com/documents/profile-fields>

Dans notre code, nous avons utilisé les informations suivantes :

- id	un identifiant unique pour le membre
- firstName	prénom du membre
- lastName	nom du membre
- educations	Une collection des établissements d'enseignement où le membre a étudié
- pictureUrl	L'URL de la photo de profil
- summary	Une zone de texte où le membre décrit son profil professionnel

Pour le code:

```
IN.API.Profile("me").fields("id", "firstName", "lastName", "educations",  
"pictureUrl", "summary").result(displayProfiles);
```

#### 4. Utilisation de JSON et AJAX (jQuery)

Remarquez que le résultat des requêtes en appelant l'objet IN retourne automatiquement un objet JSON.

Le traitement du JSON obtenu est dans la fonction displayProfiles ():

```
function displayProfiles(profiles) {
    member = profiles.values[0];
    var id=member.id;
    jQuery.ajax({
        url:'getid.php?q='+id,
        type:"get",
        success: function(data) {
            if(data.trim()=="true") { //si l'utilisateur est
dans la base de donnee
                //redirect a la page d'accueil
            } else {
                var result=JSON.stringify(member);
                var url = encodeURIComponent(result);
window.location.href="forms.php?member_linkedin="+url;
            }
        }
    }); //Fin de la partie Ajax
}
```

On note que le résultat retourné par un appel à l'API "Profile" est un objet JSON formaté. De plus, par conversion de donnée, le résultat obtenu est toujours dans un array "values". Par conséquent, la variable "member" est un objet JSON et values[0] correspond à l'objet de profil "me". Pour obtenir l'id d'un utilisateur, il faut appeler "member.id".

La fonction JSON.stringify est une fonction qui transforme un objet JSON en une chaîne de caractère (ou string) formatée. Pour plus de détail voir : <http://www.json.org/jsonfr.html>

## Quelques explications sur AJAX en utilisant JQuery

jQuery est une bibliothèque JavaScript rapide et concise, qui simplifie les déplacements dans un document HTML, la gestion des événements, l'animation, et les interactions Ajax pour le développement rapide d'applications web. Grâce à jQuery, nous pouvons rapidement développer nos applications JavaScript.

Dans notre application, pour utiliser la bibliothèque de JQuery, il suffit simplement d'ajouter dans la partie JavaScript une ligne pour importer la bibliothèque.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.4/jquery.min.js"></script>
```

Pour utiliser Ajax, c'est aussi très simple:

(Pour le détail, Allez sur <http://api.jquery.com/jquery.ajax/>)

```
jQuery.ajax({
    //Les paramètres..
});
```

## Ce que nous avons fait dans notre code:

```
jQuery.ajax({
    url: 'getid.php?q='+id,
    type: "get",
    success: function(data) {
        if(data.trim()=="true") { //si l'utilisateur est
dans la base de donnee
            //redirect a la page d'accueil
        } else {
            var result=JSON.stringify(member);
            var url = encodeURIComponent(result);
window.location.href="forms.php?member_linkedin="+url;
        }
    }
});
```

**url:** la page qui va traiter le requête AJAX.

**type:** get précise que la requête est de type "get"

**success:** Dans le cas où le serveur a bien traité la demande et a retourné un résultat, ce qui suit "success" indique ce qui va se dérouler côté client. Il est à noter que la donnée retournée est appelé "data". Si le résultat de la page "getid.php" est "true" (c'est à dire que l'id est déjà enregistré dans la base de données de Scientific-Profile), on va connecter l'utilisateur et le rediriger vers la page d'accueil.

### 2.3. VERIFICATION DE L'EXISTENCE DE L'UTILISATEUR DANS SCIENTIFIC-PROFILE

Lorsque l'on a récupéré l'id Viadeo ou LinkedIn de l'utilisateur, une requête est effectuée grâce à la page getid.php à laquelle est envoyé l'id.

L'id est recherché dans la base de données de Scientific-Profile et s'il est trouvé, le booléen "true" est renvoyé.

Notez que dans la partie AJAX de la page index.php, on a toujours une fonction qui va tester la réponse renvoyée.

Dans le code, c'est la partie de "success", qui signifie le succès de la réception de la page PHP qui traite la requête AJAX.

```
success: function(data) {
    if(data.trim()=="true") { //si l'utilisateur est
dans la base de donnee
        //redirect a la page d'accueil
    } else {
        var result=JSON.stringify(member);
        var url = encodeURIComponent(result);
window.location.href="forms.php?member_linkedin="+url;
    }
}
```

Comme vous pouvez le voir dans le code ci-dessus, si l'utilisateur est dans la base de donnée, (c'est-à-dire la réponse de la page "getid.php" est true), le programme va connecter l'utilisateur et le rediriger vers la page d'accueil de cette personne sur scientific-profile. Sinon, (c'est-à-dire si la réponse de la page "getid.php" est false ou null) le programme va utiliser la fonction stringify sur l'objet JSON et l'encoder avant de transmettre la chaîne de caractère obtenue à la page forms.php en paramètre dans l'URL.

#### Explication de **stringify** et **encodeURIComponent**

**stringify** est une fonction JSON qui va sérialiser une valeur JavaScript dans le texte JSON. JSON texte est une chaîne de caractère spécifique qui peut être traduite par JSON Parser. Cette fonction permet de faciliter la transmission de l'objet JSON, parce que du côté serveur, on a la fonction php json\_decode qui peut décoder le JSON texte.

**encodeURIComponent**: Cette fonction JavaScript permet d'encoder une chaîne de caractère pour qu'elle respecte l'encodage Web. Il permet de passer des paramètres contenant des caractères spéciaux comme le et commercial(&). Par raison de sécurité, nous avons besoin de masquer les informations de l'utilisateur dans la barre d'adresse du navigateur, nous pouvons donc aussi utiliser cette fonction à cette fin.

## 2.4. REMPLISSAGE AUTOMATIQUE DU FORMULAIRE

Si la valeur "false" est renvoyé par la page *getid.php* l'objet récupéré est envoyé vers la page *forms.php*. Ainsi, les champs du formulaire sont directement pré-remplis.

Les champs sont les suivants :

- Nom
- Prénom
- Photo de profil (URL)
- id Viadeo/LinkedIn
- Études
- Diplômes
- Présentation

Dans ce formulaire, l'utilisateur a le droit de modifier toutes ses informations sauf son id Viadeo/LinkedIn. Il a aussi la possibilité de rajouter un email et un mot de passe pour se connecter à Scientific-Profile via la connexion classique.

Étant donné qu'il est interdit d'enregistrer dans notre base de données les informations de l'API sans l'accord de l'utilisateur nous spécifions que lorsque l'utilisateur valide le formulaire, il accepte que l'on récupère ses informations via ces phrases :

*Les champs ci-dessus ont été renseignés grâce à l'API de viadeo/linkedin. En envoyant ce formulaire, vous acceptez que Scientific-Profile récupère ces données.*

Les informations de l'utilisateur ont été récupérées grâce à l'URL et mises dans une variable PHP. Nous avons utilisé le PHP ici car il est très facile de manipuler les données récupérées avec ce langage.

Dans un premier temps, la variable `member_linkedin` ou `member_viadeo` est récupérée et comme elle est sous forme d'objet JSON, elle est décodée grâce à la méthode `json_decode`.

Les données récupérées sont traitées différemment selon que l'on reçoit la variable `member_viadeo` ou la variable `member_linkedin` car les 2 Objets JSON sont différents comme nous l'avons vu précédemment.

Le formulaire est donc pré-rempli avec les données récupérées grâce à des méthodes PHP. Vous pouvez voir le formulaire à la page suivante.

## Formulaire d'inscription automatique

### Champs Optionnels

L'ajout d'un email et d'un password vous permettra de vous connecter aussi via la connection Scientific-Profile.

Dans le cas contraire vous pouvez toujours vous connecter via votre compte viadeo.

email :

email (confirmation) :

password :

password (confirmation) :

### Champs Récupérés sur viadeo



Nom :

Prénom :

Photo de profil (URL) :

id viadeo :

Cet id sera utilisé afin de vous connecter sur Scientific-Profile via la connection viadeo.

Informatique et Gestion - Ecole Polytechnique  
Universitaire De Montpellier (Ex-ISIM)

PEIP - Ecole Polytechnique Universitaire De  
Montpellier (Ex-ISIM)

BAC S SI - Lycée Jean Baptiste De Baudre

Etudes :

Diplomes :

Présentation :

Je suis en Quatrième année d'ingénieur en  
Informatique et Gestion à Polytech'Montpellier.

Les champs ci-dessus ont été renseignés grâce à l'API de viadeo. En envoyant ce formulaire, vous acceptez que Scientific-Profile récupère les données.



#### 2.4.1. RECUPERATIONS DES DONNEES

Etant donné que les deux objets LinkedIn et Viadeo sont différents, ils sont reçus dans une variable différente comme expliqué précédemment et récupérés avec une fonction PHP qui est la suivante :

```
$_REQUEST['member_linkedin'];
```

Ensuite, comme cette variable contient l'Objet JSON sous format texte, il faut convertir ce format en variable PHP grâce à la fonction `json_decode` comme ceci :

```
json_decode($member_info_sent, true);
```

Enfin, il suffit de récupérer toutes les données de l'utilisateur contenues dans la variable PHP qui est exactement constitué comme l'objet JSON. Ainsi, les 2 objets sont traités différemment. Chaque données permettant de remplir le formulaire qui est donc un forms en HTML classique.

L'extraction des données spécifique se fait simplement avec une association avec une variable comme ceci :

```
$id=$member["id"];
```

Ici, on a stocké dans la variable `$member` tout l'objet et on récupère l'id de l'utilisateur dans la variable `$id`.

Enfin, pour remplir le questionnaire, nous utilisons la fonction « echo » de PHP qui permet d'afficher le contenu d'une variable dans le forms comme ceci :

```
<input type="text" disabled="disabled" name="id" id="id" value="<?php echo $id;?>" required/>
```

#### 2.4.2. LE FORMULAIRE

Nous avons laissé la possibilité à l'utilisateur de modifier les données récupérées sauf pour le champs de l'id Viadeo/LinkedIn que nous avons bloqué grâce à l'attribut « disabled="disabled" ».

Nous avons aussi mis en option la possibilité d'obtenir un email et un mot de passe pour pouvoir se connecter via une connexion classique par la suite. Afin que cela soit sécurisé, il y a deux champs pour chacune de ces données. En effet, il est possible de se tromper en écrivant son adresse email et son mot de passe. C'est pourquoi un script JavaScript vérifie si les deux valeurs sont strictement identiques.

Voici le script de vérification :

```
<script type="text/javascript">
    function validate(form) {
        var e = form.elements;

        if(e['email'].value != e['confirm-email'].value) {
            alert('Your email do not match. Please type more
carefully.');
```

```
        return false;
    }
```

```
        if(e['password'].value != e['confirm-password'].value) {
            alert('Your passwords do not match. Please type more
carefully.');
```

```
        return false;
    }
```

```
        return true;
    }
```

```
</script>
```

De plus, le formulaire est paramétré pour envoyer les données vers une page PHP et active le script de vérification comme ceci :

```
<form method="post" action="page.php" enctype="multipart/form-data"
onsubmit="return validate(this);">
```

### 3. IMPLEMENTATION DES SHARES

L'idée de départ était de pouvoir partager sur la plupart des réseaux sociaux (Viadeo, LinkedIn, Facebook, Twitter, etc..) qu'ils soient professionnels ou non le nouveau site Scientific-Profile. Après quelques recherches, nous avons trouvé le site AddThis qui propose un outil grâce auquel on peut partager un site, une page ou même un post sur tous les réseaux sociaux grâce à un simple code en JavaScript à copier au bon endroit.

Ainsi, nous pouvons implémenter ce widget à n'importe quel niveau du site pour partager du contenu. En effet, nous pouvons l'implémenter au niveau du site, d'un profil utilisateur, mais aussi des publications.

Ce code se présente comme ceci :

```
<!-- AddThis Button BEGIN -->
<div class="addthis_toolbox addthis_default_style
addthis_32x32_style">
  <a class="addthis_button_linkedin"></a>
  <a class="addthis_button_viadeo"></a>
  <a class="addthis_button_facebook"></a>
  <a class="addthis_button_twitter"></a>
  <a class="addthis_button_google"></a>
  <a class="addthis_button_compact"></a>
  <a class="addthis_counter addthis_bubble_style" ></a>
</div>
<script type="text/javascript"
src="http://s7.addthis.com/js/250/addthis_widget.js"></script>
<!-- AddThis Button END -->
```

Cela permet d'intégrer ceci :



Notons que l'affichage est modulable et très simple d'utilisation. En effet, si on veut afficher le bouton de LinkedIn, il suffit de rajouter la balise :

```
<a class="addthis_button_linkedin"></a>
```

Ainsi, nous avons pu le paramétrer pour afficher les boutons de LinkedIn, Viadeo, Facebook, Twitter et Google+.

Tous les autres réseaux sociaux sont accessibles grâce au bouton orange avec un plus qui est rajouté grâce à cette balise :

```
<a class="addthis_counter addthis_bubble_style" ></a>
```

## 4. CREATION DE LA PAGE FACEBOOK / IMPLEMENTATION DE LA LIKEBOX

### 4.1. PAGE FACEBOOK

Pour créer une page Facebook, premièrement on visite <http://www.facebook.com/pages/create.php>.

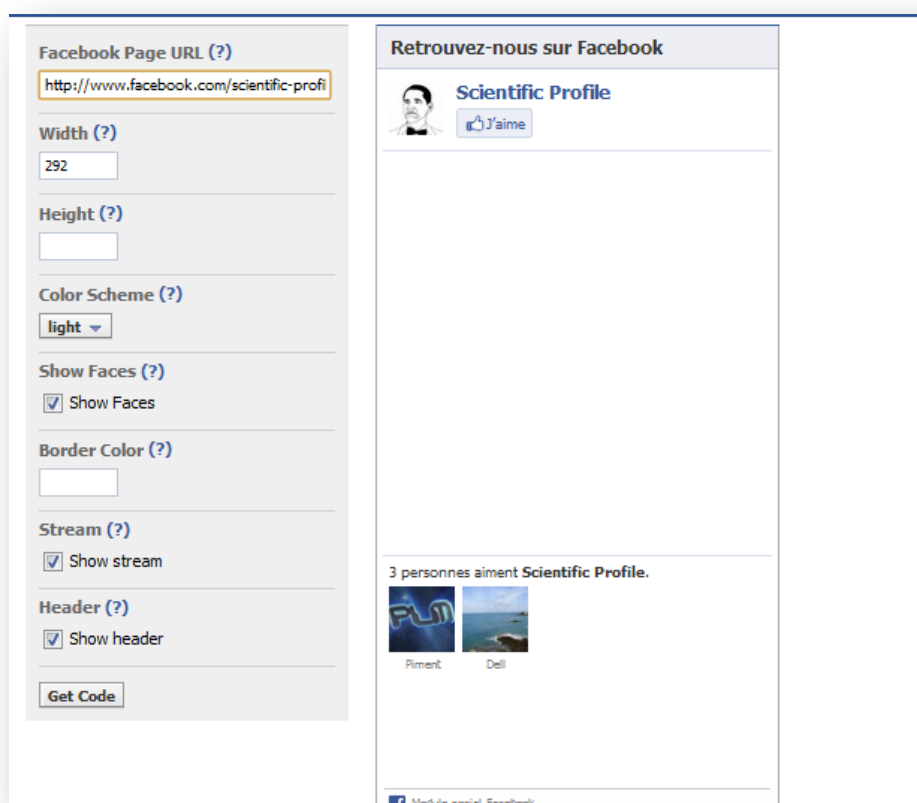
Ensuite, on crée automatiquement notre page Facebook. Pour d'autres configurations, consultez la page <http://www.squidoo.com/facebookpage>.

### 4.2. IMPLEMENTATION DE LA LIKEBOX DE FACEBOOK

Pour créer la *likebox* de Facebook, on visite premièrement l'adresse <http://developers.facebook.com>, sous la rubrique "Pour les sites web". Ensuite on va visiter le "Social Plugins" pour créer notre *likebox*.

Dans la liste de "Social Plugins", on clique sur le bouton *likebox* qui permet aux utilisateurs d'aimer votre page Facebook et afficher son flux directement à partir de votre site web.

Dans la page suivante, on a une page comme:



Le résultat va être affiché en temps réel à droite quand on change les paramètres dans la boîte à gauche. Pour générer le code, on clique sur le bouton "Get Code". Le détail de chaque champ est listé juste en bas de cette page de création.

On a plus qu'à copier / coller le code dans notre site Web, et on obtiendra la likebox comme affiché ci-dessus.

## 5. CREATION DE LA SIGNATURE DE MAIL.

Le but de cette partie est de créer une signature de mail faite grâce aux informations de la personne sur Scientific-Profile. Cette signature de mail doit être codé en html. Nous aurons aussi une deuxième version qui sera un simple texte pour prendre en compte les services de messagerie ne gérant pas l'HTML. La personne pourra ainsi avoir une signature de mail en copiant collant simplement le code qui lui sera donné sur le site.

### 5.1. LA SIGNATURE

La signature comporte les informations suivantes :

- La photo du profil
- Le laboratoire et l'institution dans lequel il travaille
- Le drapeau du pays
- Son contact (email)
- Son nombre de publications
- Le lien vers son profil Scientific-Profile
- ~~Les principaux mots clés correspondant à la personne~~ (pas implémenté pour ne pas avoir une signature trop chargée)

Afin de formater la signature, nous avons utilisé un tableau comme ceci :

Photo du profil	Prénom Nom
	Labo - Institut
	Drapeau du pays
Email	
Nombre de publication	
Lien du profil Scientific-Profile	

Vous pouvez voir l'exemple de la page proposant cette signature d'email à la page suivante.

## Signature mail



Joris Puechlong

Polytech UM2



[joris.puechlong@gmail.com](mailto:joris.puechlong@gmail.com)

Number of publications: 42

[Visit my Scientific-Profile](http://scientific-profile.com/JorisPuechlong)

## Code a copier dans votre Signature de Mail HTML

```
<html><table border="0"><tr><td ROWSPAN="3"></td><td
id="nom">Joris Puechlong</td></tr><tr><td
id="univlabo">Polytech UM2</td><tr><td></td></tr><tr><td COLSPAN="2"
id="mail"><a
href="mailto:joris.puechlong@gmail.com">joris.puechlong@gmail.
com</a></td></tr><tr><td COLSPAN="2" id="nbpubli">Number of
publications: 42</td></tr><tr><td COLSPAN="2"
id="profilelink"><a href="http://scientific-
profile.com/JorisPuechlong">Visit my Scientific-Profile</a>
</td></tr></table></html>
```

## Code a copier dans votre Signature de Mail Texte

```
Joris Puechlong
Polytech UM2
joris.puechlong@gmail.com
number of publication: 42
Visit my Scientific-Profile: http://scientific-
profile.com/JorisPuechlong
```

## 5.2. FONCTIONNEMENT

L'entreprise s'occupera de la requête nous fournissant les informations nécessaires.

Nous récupérerons un objet JSON. A partir de cela nous allons créer le code HTML et le texte simple avec toutes ces informations.

La Signature HTML sera sous forme d'un tableau HTML que voici

```
<table border="0">
  <tr>
    <td ROWSPAN="3"><img src='' id=photo WIDTH=66
HEIGHT=86 /></td>
    <td id='nom'></td>
  </tr>
  <tr>
    <td id='univlabo'></td>
  </tr>
  <tr>
    <td><img src='' alt=":" id='drapeau' /></td>
  </tr>
  <tr>
    <td COLSPAN="2" id='mail'></td>
  </tr>
  <tr>
    <td COLSPAN="2" id='nbpubli'></td>
  </tr>
  <tr>
    <td COLSPAN="2" id='profilelink'></td>
  </tr>
</table>
```

Nous modifions cette page qui est vide avec du Javascript qui va récupérer notre objet JSON, et insérer les informations dans la page.

Voici un objet JSON type.

```
var user = {
  nom : 'Puechlong',
  prenom : 'Joris',
  mail : 'joris.puechlong@gmail.com',
  univlabo : 'Polytech UM2',
  nbpubli : '42',
  profilelink : 'http://scientific-
profile.com/JorisPuechlong',
  photo : 'photo.png',
  drapeau : 'drapeau.jpg'
};
```

Pour modifier l'HTML on utilise les fonctions `document.getElementById()`. C'est pour cela que nous avons donné un Id unique à chaque balise HTML.

Voici un exemple d'utilisation :

```
document.getElementById('nom').innerHTML = user.prenom + ' ' + user.nom;
```

Le code à copier est placé dans un textarea non modifiable qui se présente ainsi :

```
<textarea name="Signaturehtml" id="signaturehtml" rows="13" cols="60"
readonly></textarea>
```

L'attribut « `readonly` » permet à l'utilisateur de copier le texte mais l'empêche de la modifier afin d'éviter de fausses manipulations qui pourraient rajouter des erreurs dans le code.

De la même manière que pour remplir le tableau en format HTML, le « `textarea` » est rempli avec tout le code sous format texte comme ceci :

```
document.getElementById('signaturehtml').value = '<html><table
border="0"><tr><td ROWSPAN="3"></td>';
document.getElementById('signaturehtml').value += '<td
id="nom">'+user.prenom+' '+user.nom+'<td></tr>';
.
.
.
```

Nous avons décomposé ce code en plusieurs parties, afin que le code soit beaucoup plus lisible et facilement modifiable par la suite.



### 5.3. LA SIGNATURE EN HTML

A partir de cette signature, nous obtenons le code HTML que la personne devra copier. Le code HTML sera le même pour tous les utilisateurs, seules les données les concernant seront différentes. Ces données sont extraites de la base de données de Scientific-Profile et récupérées grâce à un objet JSON.

Pour que tout le code affiché soit utilisable par l'utilisateur, il est mis sous forme de texte dans une balise *textarea* avec l'attribut *readonly* pour que l'utilisateur puisse le copier mais en aucun cas le modifier.

### 5.4. LA SIGNATURE TEXTE

En faisant quelques tests, nous avons remarqué que certains services de messagerie tels que Gmail ne gèrent pas l'HTML pour la signature. Nous avons donc créé une autre signature de mail sans HTML. Nous ne pouvons dans ce cas, ajouter la photo du profil ainsi que le drapeau du pays.

Celle-ci se présente donc ainsi :

```
Joris Puechlong  
Polytech UM2  
joris.puechlong@gmail.com  
number of publication: 42  
Visit my Scientific-Profile: http://scientific-profile.com/JorisPuechlong
```

Pour que l'utilisateur puisse aussi la copier, elle est implémentée exactement de la même manière que la signature HTML.

## 6. PUBLICATIONS MENDELEY

### 6.1. DEFINITION

Comme dernière tâche de notre projet, nous allons créer un module permettant d'importer les publications d'un utilisateur à partir du site Mendeley. Cela permettra d'alimenter la base de données d'Expernova.

Le but est de permettre à l'utilisateur d'importer ses publications depuis le site de Mendeley. Ainsi s'il a déjà un compte Mendeley et des documents sur ce site, il n'aura pas besoin de refaire l'étape d'ajout de ce document une énième fois.

Voici la liste des informations que nous devons récupérer sur les documents :

- Le titre
- Le résumé
- Le Lien vers le document
- Les mots clés
- Les auteurs
- L'année

### 6.2. CHOIX ET TECHNOLOGIES

Pour l'implémentation des fonctionnalités désirées, nous avons choisis d'utiliser le PHP. En effet contrairement aux APIs de Viadeo et LinkedIn, l'API de Mendeley n'est pas autant développée et la connexion avec l'API ne se fait pas simplement avec l'import d'un script JavaScript et l'appel d'une fonction.

Nous aurions pu réaliser cette partie en Python comme c'est proposé sur le site de Mendeley, mais étant donné que nous n'avons aucune base en Python et que le temps qu'il nous restait n'était pas suffisant, nous avons utilisé le PHP.

Comme la plupart des connexions avec les APIs, c'est le protocole OAuth qui va être utilisé. C'est le même protocole qui est utilisé par LinkedIn et Viadeo, mais celui-ci était complètement caché derrière l'utilisation du JavaScript. C'est cette implémentation qui a été notre principale difficulté dans cette partie du projet.

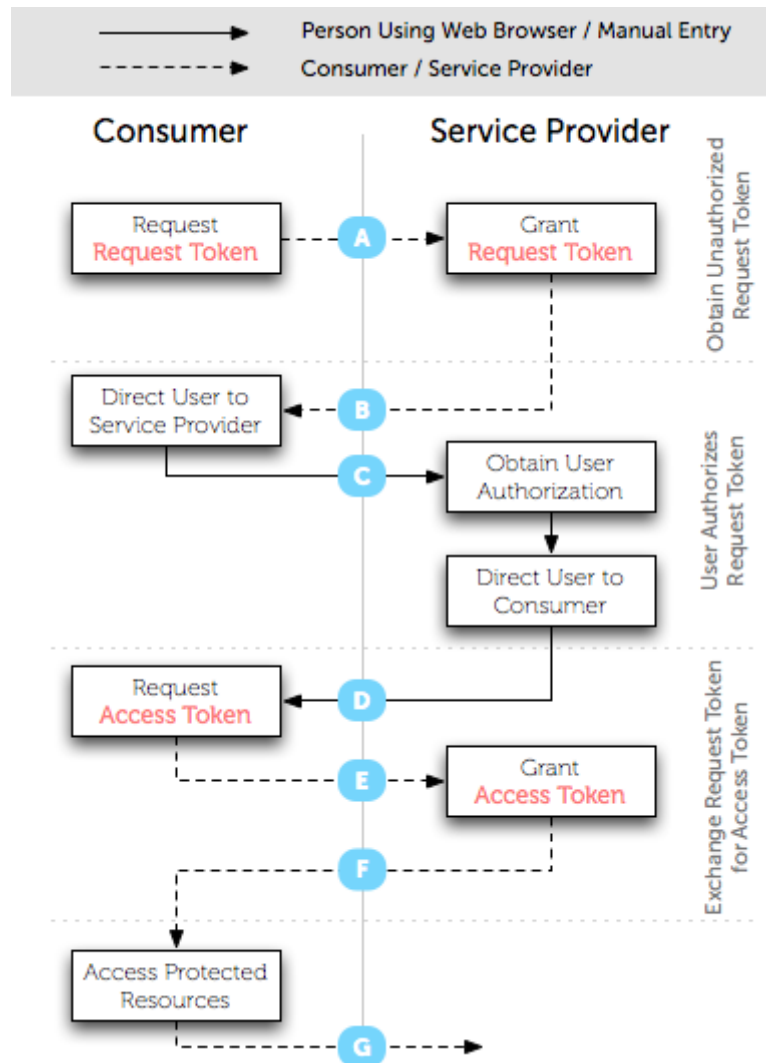
Pour pouvoir réaliser notre travail, nous avons utilisé un serveur local sous Unix (Ubuntu) sur lequel nous avons installé OAuth. En effet, pour que cela puisse fonctionner, il faut que OAuth soit installé sur le serveur.

### 6.3. CONNEXION A L'API MENDELEY

De la même manière que Viadeo et LinkedIn, pour pouvoir se connecter à l'API, nous avons récupéré un CONSUMER\_KEY et un CONSUMER\_SECRET. A partir de là, il a fallu développer tout le processus de connexion à l'API et d'identification de l'utilisateur pour pouvoir récupérer toutes ses données. Afin d'avoir le code le plus clair possible, nous avons séparé notre code en plusieurs pages PHP.

Chaque requête à l'API Mendeley nous renvoie un objet sous format JSON.

Ci-dessous, voici un schéma représentant les étapes pour la connexion à l'API :



Dans un premier temps, il faut créer un Objet OAuth avec le OAUTH\_CONSUMER\_KEY et le OAUTH\_CONSUMER\_SECRET. L'objet est créé ainsi :

```
$oauth = new OAuth($OAUTH_CONSUMER_KEY,$OAUTH_CONSUMER_SECRET);
```


Grâce à cet objet, une requête de Token est envoyée à l'API et l'API nous renvoie un OAUTH\_TOKEN et un OAUTH\_TOKEN\_SECRET, une sorte d'identifiant pour la session. (A & B sur le schéma). La requête est effectuée ainsi :

```
$oauth->getRequestToken($request_token_url, $callback_url);
```

Où `$request_token_url` est l'URL de demande de l'API et `$callback_url` est la page PHP où les données vont être réceptionnées.

Les deux dernières données sont associées à l'objet OAuth et une requête est envoyée à l'API pour identifier l'utilisateur et avoir son accord grâce aux fenêtres pop-up générées par l'API de Mendeley. Cette requête se traduit par un simple lien internet avec le OAUTH\_TOKEN en paramètre comme ci-dessous. (C & D)

```
<a href="http://api.mendeley.com/oauth/authorize/?oauth_token=?php echo $request_token; ?>">click here to import your documents from Mendeley</a>
```




**An application would like to connect to your account**

Sign in below to accept or reject this request.

E-mail:  ☐ Remember me

Password:  [Forgot your password?](#)

**Sign in**



**An application would like to connect to your account**

The application is requesting the ability to access and update data from your **Mendeley** account.

joris.jojo@gmail.com

**Accept**

Enfin, une dernière requête est effectuée afin d'obtenir une autorisation d'accès à la session de l'utilisateur. La requête est effectuée ainsi :

```
$oauth->getAccessToken ("http://api.mendeley.com/oauth/access_token/");
```

Maintenant que l'étape de la connexion est finie, nous pouvons réaliser les requêtes que nous voulons pour obtenir les données souhaitées.

#### 6.4. IMPORTATION DES DOCUMENTS.

Une fois l'étape de connexion faite, la liste des documents est récupérée en deux étapes. La première consiste à lister les documents de l'utilisateur grâce à cette requête :

```
$oauth->fetch ("http://api.mendeley.com/oapi/library/documents/authored/");
```

Malheureusement, cette requête ne donne pas assez d'informations sur les documents. C'est pourquoi une seconde requête est effectuée sur chaque document pour obtenir les informations désirées. Cette requête est faite en fonction de l'ID du document obtenu avec le premier résultat. Voici la requête :

```
$oauth->fetch ("http://api.mendeley.com/oapi/library/documents/". $document);
```

Où \$document est l'ID du document.

Dans l'importation des documents, nous laissons le choix à l'utilisateur d'importer les documents de son choix sur son profil Scientific-Profile. C'est pourquoi dans un premier temps, l'utilisateur se retrouve face à un formulaire en HTML où il coche les documents qu'il souhaite obtenir.

### Select the documents you want to import

☐ le document insipide

Abstract: je suis le petit rÃ©sumÃ© du libre

☐ monArticle

Abstract: mon abstract

Ce formulaire est un ensemble de cases à cocher avec le titre du document et son résumé. Pour conserver la totalité des informations, la valeur de la checkbox est constituée de tout l'objet JSON sous format texte.

Lorsque l'utilisateur a fini de sélectionner les documents et valider le formulaire, la liste des documents est envoyée vers une page PHP qui traite les objets JSON sous format texte. A ce niveau-là, il faut savoir que les objets JSON sont constitués tels que nous les avons récupérés sur Mendeley. Le but de cette page est de créer les mêmes documents mais dans un format compatible pour Scientific-Profile.

Pour les attributs simples comme le titre ou le résumé, cette étape est facile à réaliser, mais en ce qui concerne le type de document (thèse, livre, etc...) il faut faire une corrélation puisque Mendeley met du texte comme attribut (exemple : « book ») alors que Scientific-Profile utilise des nombres (exemple : 5 qui correspond à book).

Dans la page qui traite les objets JSON, il faut 2 étapes : Une étape pour décoder l'objet et une étape pour le recoder. Dans l'étape pour décoder, on utilise la fonction PHP «json\_decode », et on met le résultat sous la forme d'un tableau (array) en ajoutant un paramètre «true » à la fin de cette fonction.

```
$mendeley = json_decode($mendeley_string, true) ;
```

*Pour la récupération de titre par exemple,*

```
$mendeley = json_decode($mendeley_string, true) ;  
$title_sp = $mendeley["title"] ;
```

*Pour l'étape de recodage, on a besoin d'utiliser la fonction PHP json\_encode () et c'est aussi très simple :*

```
$result_JSON = json_encode($array_sp)  
$title_sp = $mendeley["title"];
```

*Notons que dans le corps de json\_encode on met un array. La déclaration et l'affectation d'un array est ainsi :*

```
$array_sp = array('title'=>NULL, 'doc_abstract'=>'My description'..) ;  
  
Ou  
  
$array_sp = new array() ;  
  
$array_sp[] = 'title' ;
```

*Le JSON String pour Scientific Profile :*

```
{"title":"Thesis Test title","doc_abstract":null,"keywords":"thesis, polytech, php, France","year":"2012","type":0,"source":"http://www.thegooddeedmovement.com","authors":[{"id_e":null,"forename":"Chenyang","lastname":"GAO","affiliation":null,"id_a":null,"related":[{"name":null,"id":null}]}],{"id_e":null,"forename":"Blunt","lastname":"James","affiliation":null,"id_a":null,"related":[{"name":null,"id":null}]}], "domains":null}
```