

Fil d'actualité

Honoré Hounwanou

Fil d'actualité

Honoré Hounwanou

©2015 Honoré Hounwanou

Table des matières

Introduction	1
1. La logique	2
1.1 Préparer le terrain	2
1.2 Premies pas	5
1.3 La requête SQL	8
1.4 Test	12
2. La vue	16
2.1 Modification du fichier _micropost.php	16
2.2 Test	16
2.3 Correction d'un léger problème	20
Conclusion	23

Introduction

Mama Miya,

Hello guys, heureux de vous retrouver dans cette nouvelle vidéo des [TEACHERS DU NET¹](#) :).

Au stade actuel, nous avons la possibilité de pouvoir nous faire des amis (envoyer une demande d'amitié, accepter une demande d'amitié...), mais pour l'instant, si on y regarde d'un peu plus près cela ne sert quasiment à rien !

Ce que je veux dire, c'est que nous n'avons aucun moyen de voir les publications nos différents amis. Je vous propose donc dans ce tutoriel de rectifier le tir en concevant un petit fil d'actualité. Ce sera une version miniature du fil d'actualité de Facebook.

Ne vous inquiétez surtout pas, ce sera assez super facile (comme d'habitude :)). Ne perdons donc plus de temps et découvrons si “**Alemke**” raconte de mauvaises choses sur nous.



Attention !

Lorsqu'un code ne peut pas tenir sur une ligne, vous avez un anti-slash qui est rajouté avant de passer à la ligne suivante. Pensez donc à enlever ce anti-slash si vous faites du copier-coller.

¹<http://youtube.com/hounwanou1993>

1. La logique

La page chargée d'afficher les différentes publications est la page de profil. Il va s'en dire que nous devons modifier deux fichiers :

- Le fichier `profile.php`, afin de récupérer cette fois-ci non seulement les publications de l'utilisateur dont on est sur la page de profil mais également les publications de ses amis. Quand je parle de publications, je fais bien sûr référence aux microposts.
- Le fichier `views/profile.view.php` afin de pouvoir harmoniser un tant soit peu les variables utilisées au cas où nous avons quelques erreurs.

Dans ce premier chapitre, nous allons nous intéresser à la logique. Il s'agira d'écrire la requête SQL adéquate pour la génération du fil d'actualité. Une fois notre requête SQL fonctionnelle, il suffira d'adapter la vue en conséquence. Nous nous chargerons de la vue dans le second chapitre.

Maintenant que tout a été planifié, les choses sérieuses peuvent enfin démarrer.

1.1 Préparer le terrain

Nous allons dans un premier temps, faire en sorte que tous les utilisateurs soient amis avec eux-mêmes. Rappelez vous qu'on ne l'avait pas fait pour nos utilisateurs fictifs.

Connectez vous premièrement au serveur MySQL :

```
1 mysql -u root -p
```

Puis à la base de données :

```
1 use boom;
```

- Suppression de toutes les données présentes au niveau de la table `microposts`

```
1 DELETE FROM microposts
```

- Suppression de toutes les données présentes au niveau de la table `friends_relationships`

```
1 DELETE FROM friends_relationships;
```

- Suppression de toutes les données présentes au niveau de la table codes

```
1 DELETE FROM codes;
```

- Suppression de toutes les données présentes au niveau de la table notifications

```
1 DELETE FROM notifications;
```

- Suppression de toutes les données présentes au niveau de la table auth_tokens

```
1 DELETE FROM auth_tokens;
```

- Suppression de toutes les données présentes au niveau de la table users

```
1 DELETE FROM users;
```



Attention !

Il est important de respecter l'ordre de suppression. La table users doit être supprimée en dernière position étant donné que les autres tables en dépendent (via les clés étrangères).

Modifions ensuite le fichier seed/users.php avec ce nouveau contenu :

```
1 <?php
2
3 require '../config/database.php';
4
5 require '../vendor/autoload.php';
6
7 $faker = Faker\Factory::create();
8
9
10 for ($i=1; $i <= 30 ; $i++) {
11
```

```

12     $q = $db->prepare('INSERT INTO users(name, pseudo, email, password, active,
13                          created_at, city, country, sex, available_for_hiring, bio)
14                          VALUES(:name, :pseudo, :email, :password, :active,
15                          :created_at, :city, :country, :sex,
16                          :available_for_hiring, :bio)');
17
18     $q->execute([
19         'name' => $faker->unique()->name,
20         'pseudo' => $faker->unique()->userName,
21         'email' => $faker->unique()->email,
22         'password' => password_hash('123456', PASSWORD_BCRYPT),
23         'active' => 1,
24         'created_at' => $faker->date().' '.$faker->time(),
25         'city' => $faker->city,
26         'country' => $faker->country,
27         'sex' => $faker->randomElement(['H', 'F']),
28         'available_for_hiring' => $faker->randomElement([0, 1]),
29         'bio' => $faker->paragraph()
30     ]);
31
32     $id = $db->lastInsertId();
33
34     $q = $db->prepare("INSERT INTO friends_relationships(user_id1, user_id2, sta\
35 tus)
36                          VALUES(?, ?, ?)");
37     $q->execute([$id, $id, '2']);
38 }
39
40 echo 'Users added!!!';

```

Ouvrez ensuite votre navigateur et rendez vous à l'adresse <http://localhost:8000/seed/users.php>. 30 utilisateurs seront créés et seront tous amis avec eux mêmes.

Vous pouvez vérifier en tapant les requêtes :

```

1 SELECT * FROM users;
2 SELECT * FROM friends_relationships;

```

Essayer maintenant de vous connecter avec un utilisateur et faites vous 2 ou trois amis. Pensez à publier des posts avec le compte de vos fameux amis. Cela nous servira pour les tests ultérieurs.

1.2 Premies pas

Ouvrez à présent le fichier `profile.php`. Son contenu devrait ressembler à quelque chose de ce genre :

```

1  <?php
2  session_start();
3
4  require("includes/init.php");
5  include('filters/auth_filter.php');
6
7
8  if(!empty($_GET['id'])){
9      //Recuperer les infos sur l'user en bdd en utilisant son id
10     $user = find_user_by_id($_GET['id']);
11
12     if(!$user){
13         redirect('index.php');
14     } else {
15         $q = $db->prepare('SELECT id, content, created_at FROM microposts
16                             WHERE user_id = :user_id
17                             ORDER BY created_at DESC');
18
19         $q->execute([
20             'user_id' => $_GET['id']
21         ]);
22
23         $microposts = $q->fetchAll(PDO::FETCH_OBJ);
24     }
25
26 } else {
27     redirect('profile.php?id='.$get_session('user_id'));
28 }
29
30 require("views/profile.view.php");

```

Comme vous pouvez le voir, la requête SQL actuelle permet de récupérer uniquement les microposts de l'utilisateur dont on est sur la page de profil. Si nous sommes donc sur la page de profil de `mercuryseries`, seuls les microposts de `mercuryseries` seront affichés. Pareil pour `alemke` et ainsi de suite.

Quelle est donc la fameuse requête SQL qu'il va nous falloir écrire afin de récupérer à la fois les microposts de l'utilisateur dont on est sur la page de profil mais également ceux de ses amis ?

Bonne question !

Ce que je me dis, c'est que nous aurons besoin de relier trois tables :

- La table **microposts**
- La table **users**
- et la table **friends_relationships**,

car pour afficher des microposts, il va falloir aller les chercher au niveau de la table microposts et pour savoir lesquelles appartiennent nos amis, il va falloir dans un premier temps connaître nos amis et un ami reste un simple utilisateur :).

J'espère que tout cela ne vous semble pas confus. Le procédé que j'ai eu à faire est tout simple.

Je me suis premièrement posé cette question :

Qu'est-ce que je souhaite faire ?

Et la réponse est :

Je souhaite créer un fil d'actualité qui va permettre d'afficher à la fois les **microposts** de l'**utilisateur** dont on est sur la page de profil mais également ceux de ses **amis**.

Après cette étape, tout ce que j'ai eu à faire c'est recenser les noms de tables qui se retrouvent dans la réponse précédente.

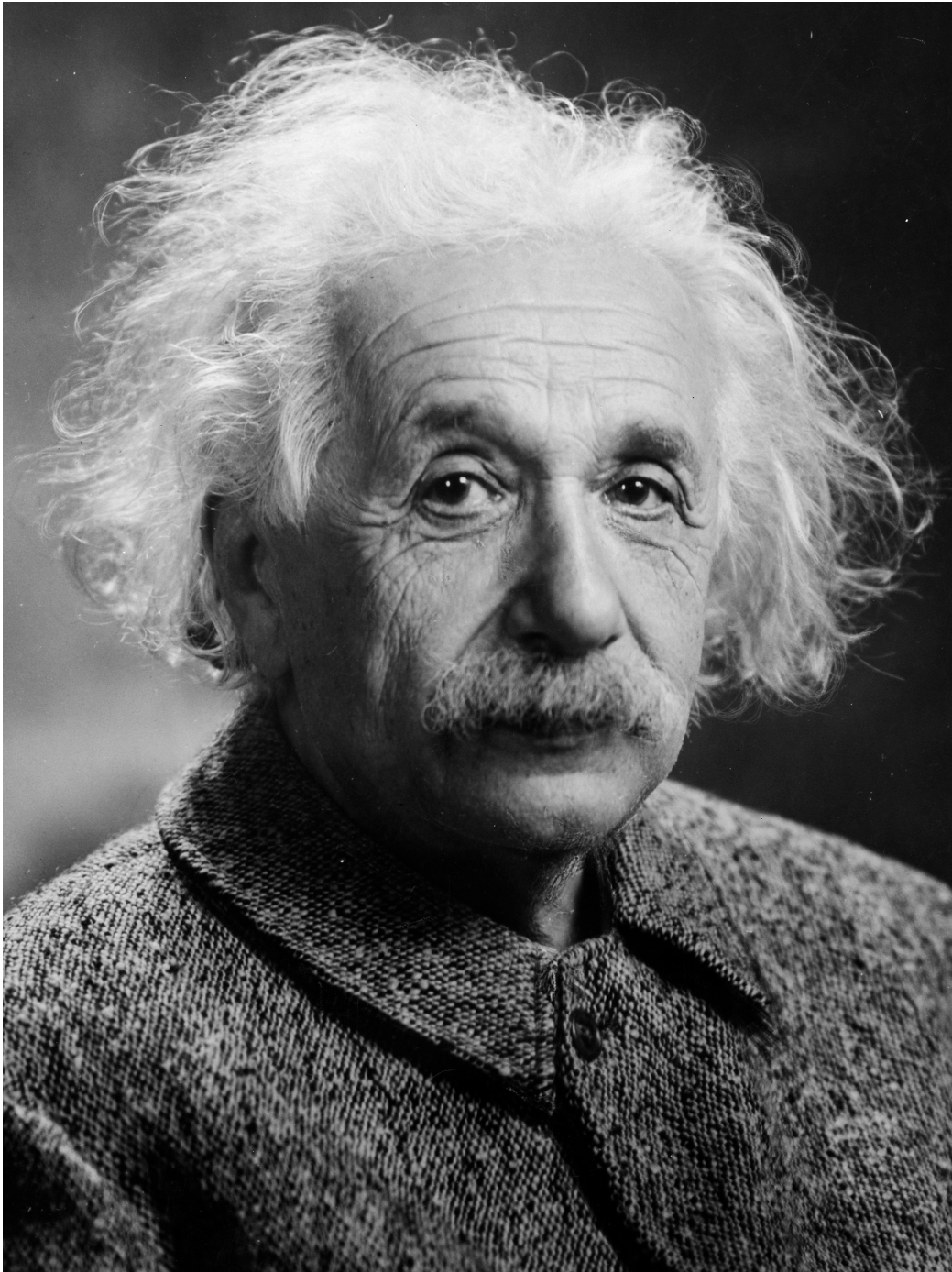
On obtient donc comme résultat :

- microposts
- utilisateurs (dans notre cas cela équivaut à la table users)
- amis (dans notre cas cela équivaut à la table friends_relationships)

Et voilà ! Bingo !

Comme vous pouvez le voir, pas besoin d'être [Albert Einstein](https://fr.wikipedia.org/wiki/Albert_Einstein)² pour arriver à cette conclusion :).

²https://fr.wikipedia.org/wiki/Albert_Einstein



Albert Einstein

La vraie question, c'est comment relier ces tables ?

1.3 La requête SQL

Je vais vous présenter un exemple de requête SQL que nous pouvons écrire et vous l'expliquer pas à pas.

```
1  SELECT U.id user_id, U.pseudo, U.email, U.avatar, M.id m_id, M.content, M.create\
2  d_at
3  FROM users U, microposts M, friends_relationships F
4  WHERE M.user_id = U.id
5
6  AND
7
8  CASE
9      WHEN F.user_id1 = :user_id
10     THEN F.user_id2 = M.user_id
11
12     WHEN F.user_id2 = :user_id
13     THEN F.user_id1 = M.user_id
14  END
15
16  AND F.status > 0
17
18  ORDER BY M.created_at DESC
19
20  -- Juste un commentaire:
21  -- :user_id sera remplacé par $_GET['id']
```

Prenez le temps de lire cette requête SQL afin de vous donner une première idée de ce à quoi elle est censée servir.

J'avoue que pour quelqu'un qui débute, cela peut sembler un tout petit compliqué raison pour laquelle j'ai choisi de ne pas expliquer cette requête SQL. On va juste faire du copier-coller. Vous verrez que cela fonctionne. C'est l'essentiel n'est-ce pas :) ?

Bien sûr je rigole. Prenez toujours l'habitude de comprendre le code que vous utilisez. Il ne faudrait pas faire uniquement du copier-coller. Lorsque nous avons voulu mettre en place notre système de "Remember me ou Garder ma session active", vous avez vu qu'on a eu à se servir d'un article externe. Mais bien avant de copier-coller le code, nous avons tenter de comprendre dans l'ensemble ce à quoi il servait, ce qui nous permis de facilement le modifier à notre guise.



Attention !

Dans certains cas, le copier-coller sans nécessité de comprendre le code source est tout à fait compréhensible. Prenons le cas des librairies présentes au niveau de notre fichier `composer.json`. Nous les utilisons, mais nous n'avons aucune idée de comment fonctionne le code source derrière. Si plus tard nous souhaitons modifier quelque chose au niveau de ces librairies ou tout simplement les améliorer (rappelez vous rien n'est parfait), à ce moment là, nous pouvons décider de nous pencher sérieusement sur le code source de ces librairies.

Trève de bavardage ! Je vois que je me suis lancé dans un mini roman sans pour autant expliquer ce à quoi servait notre fameuse requête.

```
1  SELECT U.id user_id, U.pseudo, U.email, U.avatar, M.id m_id, M.content, M.create\
2  d_at
3  FROM users U, microposts M, friends_relationships F
4  WHERE M.user_id = U.id
5
6  AND
7
8  CASE
9      WHEN F.user_id1 = :user_id
10     THEN F.user_id2 = M.user_id
11
12     WHEN F.user_id2 = :user_id
13     THEN F.user_id1 = M.user_id
14 END
15
16 AND F.status > 0
17
18 ORDER BY M.created_at DESC
19
20 -- Juste un commentaire:
21 -- :user_id sera remplacé par $_GET['id']
```

Comme vous l'aurez deviné, cette requête SQL permet de sélectionner tous les microposts de l'utilisateur dont on est sur la page de profil de même que ceux de ses amis.

Vous remarquerez que notre hypothèse précédente s'avère vérifiée dans la mesure où les trois tables `users`, `friends_relationships` et `microposts` ont bel et bien été mises en relation.

Découpons à présent notre requête par morceaux. Ne soyons pas gourmands :).

```
1 SELECT U.id user_id, U.pseudo, U.email, U.avatar, M.id m_id, M.content, M.create\  
2 d_at  
3 FROM users U, microposts M, friends_relationships F  
4 WHERE M.user_id = U.id
```

- Dans un premier temps, nous sélectionnons les identifiants, les pseudonymes, les adresses emails, les avatars de tous les utilisateurs et le contenu, la date de création de tous les microposts.

En lieu et place d'avoir à me fatiguer à taper `users.id`, `users.pseudo`, `users.avatar`, `microposts.content`, `microposts.created_at`, nous avons créer des alias.

En mettant

```
1 users U, microposts M, friends_relationships F
```

C'est comme si on disait qu'au niveau de cette requête, écrire `users` ou `U` sont équivalents, de même que `microposts` et `M` ou encore `friends_relationships` et `F`.

Plus explicitement, on aurait pu mettre :

```
1 users as U, microposts as M, friends_relationships as F
```

J'ai juste rajouté le mot-clé `as` pour dire explicitement que `U` est un alias de `users`, `M` un alias de `microposts` et `F` un alias de `friends_relationships`.

En effet, le mot-clé `as` est facultatif. Si vous ne le mettez pas, MySQL implicitement saura que vous souhaitez créer des alias. C'est donc pour cette raison que je n'avais pas mis ce fameux mot-clé `as`.

Ce que je veux dire c'est que :

```
1 SELECT U.id user_id, U.pseudo, U.email, U.avatar, M.id m_id, M.content, M.create\  
2 d_at  
3 FROM users U, microposts M, friends_relationships F  
4 WHERE M.user_id = U.id
```

et

```
1 SELECT U.id as user_id, U.pseudo, U.email, U.avatar, M.id as m_id, M.content, M.\
2 created_at
3 FROM users as U, microposts as M, friends_relationships as F
4 WHERE M.user_id = U.id
```

sont deux **requêtes identiques**.

Je vais pour ma part garder la seconde requête avec le mot-clé **as** car je préfère être toujours explicite.

Pas besoin donc d'expliquer `U.id as user_id` et `M.id as m_id`, j'ai juste créer deux alias `user_id` et `m_id`.

Parfait!

- Ensuite on filtre les résultats afin d'avoir uniquement les microposts où l'utilisateur dont l'identifiant est égal à `$_GET['id']` (donc l'utilisateur dont on est sur la page) a une entrée au niveau de la table `friends_relationships` avec l'utilisateur sur lequel on est sur l'id dans la boucle.

```
1 AND
2
3 CASE
4     WHEN F.user_id1 = :user_id
5     THEN F.user_id2 = M.user_id
6     WHEN F.user_id2 = :user_id
7     THEN F.user_id1 = M.user_id
8 END
```

- Puis, on s'assure que la dite entrée au niveau de la table `friends_relationships` confirme que les deux utilisateurs sont amis.

```
1 AND
2 F.status > 0
```

Quand est-ce que deux utilisateurs sont amis ?



C'est tout simplement lorsqu'au niveau du champ `status` on a soit **1** ou soit **2**. Rappelez vous chaque utilisateur est ami avec lui-même, ce qui est symbolisé par le chiffre **2**. C'est pour cette raison qu'on se dit que deux utilisateurs sont amis si la valeur de `status` est supérieur à **0**.

- Pour terminer, on ordonne les microposts du plus récent au plus ancien en se servant de la date de création.


```
25
26             WHEN F.user_id2 = :user_id
27             THEN F.user_id1 = M.user_id
28         END
29
30         AND F.status > 0
31         ORDER BY M.id DESC");
32
33     $q->execute([
34         'user_id' => $_GET['id']
35     ]);
36
37     $microposts = $q->fetchAll(PDO::FETCH_OBJ);
38 }
39
40 } else {
41     redirect('profile.php?id=' . get_session('user_id'));
42 }
43
44 require("views/profile.view.php");
```


Ouvrez maintenant votre navigateur et rendez vous au niveau de la page de profil d'un utilisateur qui a au moins des amis.

**mercuryseries**

⌚ il y a environ 17 minutes  Notice: Undefined property: stdClass::\$id in C:\xampp\htdocs\tutos\social_network\partials_micropost.php on line 9 Call Stack #TimeMemoryFunctionLocation 1 0.0030131080{main}()...\profile.php:0 20.0300216768require('C:\xampp\htdocs\tutos\social_network\views\profile.view.php')...\profile.php:45 30.0340221960include('C:\xampp\htdocs\tutos\social_network\partials_micropost.php')...\profile.view.php:84 ">  Supprimer

Je suis elsa38

**mercuryseries**

⌚ il y a environ 17 minutes  Notice: Undefined property: stdClass::\$id in C:\xampp\htdocs\tutos\social_network\partials_micropost.php on line 9 Call Stack #TimeMemoryFunctionLocation 1 0.0030131080{main}()...\profile.php:0 20.0300216768require(

Bug affichage du fil d'actualité

Il nous reste à gérer deux petits problèmes.

- Le lien de suppression est invalide, ce qui cause cette erreur.
- De plus, pour l'instant, c'est le pseudonyme de l'utilisateur dont on est sur la page qui est affiché pour chacun des microposts (Pareil pour l'avatar). Pourquoi me direz vous ? Eh bien, parce que c'est ce qu'on a demandé de faire :).

Si vous ouvrez le fichier `partials/_micropost.php`, vous verrez qu'on affiche à chaque fois les attributs pseudo, avatar et email de l'objet `$user`.

```

1 <article class="media status-media">
2   <div class="pull-left">
3     pseudo ?>" class="media-object avatar-xs">
5   </div>
6   <div class="media-body">
7     <h4 class="media-heading"><?= e($user->pseudo); ?></h4>
8     ...
9   </div>
10 </article>

```

A quoi correspond cet objet `$user` ? Pour le savoir, retournons au niveau du fichier `profile.php`

```
1 $user = find_user_by_id($_GET['id']);
```

Il correspond à l'utilisateur dont on est sur la page de profil. Tout s'explique donc.

Il nous faudra donc modifier ces valeurs de manière appropriée. Pour cela, je vous donne rendez-vous au prochain chapitre.

2. La vue

Il va nous falloir modifier le fichier `partials/_micropost.php` afin qu'il puisse afficher les bonnes informations. Ce sera assez simple vu que les informations en question ont déjà été récupérées.

2.1 Modification du fichier `_micropost.php`

Ouvrez le fichier `partials/_micropost.php` et remplacez son contenu par le suivant :

```
1 <article class="media status-media">
2     <div class="pull-left">
3         pseudo ?>" class="media-object avatar-\
5 xs">
6     </div>
7     <div class="media-body">
8         <h4 class="media-heading"><?= e($micropost->pseudo); ?></h4>
9         <p><i class="fa fa-clock-o"></i> <span class="timeago" title="<?= $micro\
10 post->created_at ?>"><?= $micropost->created_at ?></span>
11         <a data-confirm="Voulez-vous vraiment supprimer cette publication ?" href\
12 f="delete_micropost.php?id=<?= $micropost->m_id ?>"><i class="fa fa-trash"></i> \
13 Supprimer</a></p>
14         <?= nl2br(replace_links(e($micropost->content))); ?>
15     </div>
16 </article>
```

On ne fait que remplacer tous les `$user` par `$micropost` et `$micropost->id` par `$micropost->m_id` qui représente l'identifiant du micropost. Et le tour est joué !

2.2 Test

Ouvrez de nouveau votre navigateur et testez, vous verrez que tout fonctionne à la perfection.

**angela75**
🕒 il y a environ 24 jours [Supprimer](#)
Je suis angela75

**alemke**
🕒 il y a environ 24 jours [Supprimer](#)
Je suis Alemke

**angela75**
🕒 il y a environ 24 jours [Supprimer](#)
Je suis cool - Angela75

**mercuryseries**
🕒 il y a environ 2 mois [Supprimer](#)
Oh life is goog :)

Fil d'actualité fonctionnel

Je suis actuellement connecté avec **mercuryseries**. Je vais envoyer une demande d'amitié à **chloe74**.



Votre demande d'amitié a été envoyée avec succès!

Profil de chloe74 (1 ami)



Demande d'amitié déjà envoyée.

Annuler la demande

chloe74

katelin.russel@mertz.com

📍 Mullerfurt - Korea

[Voir sur Google Maps](#)

🚫 Non disponible pour emploi

Petite biographie de Malinda Towne

Tenetur et est ipsam aut porro. Minima deserunt sed inventore cum ipsa a. Quod porro incidunt cupiditate. Eaque necessitatibus ea quisquam repudiandae sunt. Ratione ipsum aspernatur aliquam doloribus officiis.



alemke

🕒 il y a environ 21 jours 🗑 Supprimer

Je tess



alemke

🕒 il y a environ 21 jours 🗑 Supprimer

Je teste



alemke

🕒 il y a environ 24 jours 🗑 Supprimer

Je suis Alemke

Envoi d'une demande d'amitié à chloe74

Puis, je vais me connecter avec chloe74, afin d'accepter la demande d'amitié précédemment envoyée par mercuryseries.



🔔 (1)

Alors quoi de neuf?

Publier




alemke

🕒 il y a environ 21 jours 🗑 Supprimer

Je tess

chloe74 reçoit une notification


Vos notifications

 **mercuryseries** vous a envoyé une demande d'amitié il y a environ une minute. [Accepter](#) [Decliner](#)

chloe74 affiche ses notifications

Vous etes a present ami avec cet utilisateur!

Profil de mercuryseries (4 amis)





[Retirer de ma liste d'amis](#)

mercuryseries
mercuryseries@gmail.com
[Voir sur Google Maps](#)

Non disponible pour emploi

Petite biographie de Mercury Series
Aucune biographie pour le moment...

 **alemke**
il y a environ 21 jours [Supprimer](#)
Je tess

 **alemke**
il y a environ 21 jours [Supprimer](#)
Je teste

chloe74 accepte la demande d'amitié de mercuryseries

Cela étant fait, je vais maintenant poster une nouvelle publication avec le compte de chloe74.

Boom Social Network Liste des utilisateurs

Profil de chloe74 (2 amis)



chloe74
katelin.russel@mertz.com
Mullerfurt - Korea
[Voir sur Google Maps](#)

Non disponible pour emploi

Petite biographie de Malinda Towne
Tenetur et est ipsam aut porro. Minima deserunt sed inventore cum ipsa

 **alemke**
il y a environ 21 jours [Supprimer](#)
Je tess

Je suis chloe et j'aime mercuryseries :)

[Publier](#)

chloe74 poste une nouvelle publication

Maintenant que chloe74 est dans la liste d'amis de mercuryseries, il devrait normalement voir la publication de chloe74 au niveau de son fil d'actualité.



La publication de chloe74 apparait sur le fil d'actualité de mercuryseries

Comme vous pouvez le voir tout fonctionne comme on l'espérait.

2.3 Correction d'un léger problème

Au niveau du fil d'actualité, lorsque mercuryseries est connecté, on lui affiche un lien de suppression sur les microposts des autres utilisateurs.

**angela75**
🕒 il y a environ 24 jours [Supprimer](#)
Je suis angela75

**alemke**
🕒 il y a environ 24 jours [Supprimer](#)
Je suis Alemke

**angela75**
🕒 il y a environ 24 jours [Supprimer](#)
Je suis cool - Angela75

**mercuryseries**
🕒 il y a environ 2 mois [Supprimer](#)
Oh life is goog :)

Fil d'actualité fonctionnel

Même si le micropost ne sera pas supprimé lorsqu'il cliquera sur le lien de suppression, étant donné qu'il n'est pas l'auteur du micropost (rappelez vous qu'on avait eu à faire cette vérification dans le tutoriel précédent avant toute suppression), il serait plus prudent de ne pas afficher de lien de suppression afin de ne pas donner des idées noires à un utilisateur malicieux.

Pour régler ce léger problème, ce sera très facile. Ouvrez le fichier `partials/_micropost.php` et rajouter cette condition autour du lien de suppression :

```
1 <?php if($micropost->user_id == get_session('user_id')): ?>  
2 <?php endif; ?>
```


Ce qui donne :



```


1 <article class="media status-media">
2   ...
3   <div class="media-body">
4     ...
5     <?php if($micropost->user_id == get_session('user_id')): ?>
6     <p>...
7       <a data-confirm="Voulez-vous vraiment supprimer cette publication ?"\
8 href="delete_micropost.php?id=<?= $micropost->m_id ?>"><i class="fa fa-trash"><\
9 /i> Supprimer</a></p>
10     <?php endif; ?>
11   </p>
12   <?= nl2br(replace_links(e($micropost->content))); ?>
13 </div>
14 </article>


```

Et voilà tout est bien qui finit bien.


angela75
 ⌚ il y a environ 24 jours
 Je suis angela75


alemke
 ⌚ il y a environ 24 jours
 Je suis Alemke


angela75
 ⌚ il y a environ 24 jours
 Je suis cool - Angela75


mercuryseries
 ⌚ il y a environ 2 mois [Supprimer](#)
 Oh life is goog :)

Fil d'actualité fonctionnel

Conclusion

Houra ! Chacun de nos utilisateurs a maintenant la possibilité de pouvoir voir les publications de ses amis. Il nous a fallu juste réfléchir pendant quelques minutes sur la requête SQL à écrire. A part cela, comme vous avez pu le voir, il n'y avait vraiment rien de bien compliqué.

Si vous avez néanmoins quelques incompréhensions au niveau de certaines parties, n'hésitez surtout pas à m'envoyer un mail à mercuryseries@gmail.com, je me ferai un plaisir de vous répondre.

So thank you guys for watching. Don't forget to subscribe and have a nice day. Peace !