Système de notifications en PHP

Honoré Hounwanou

Système de notifications en PHP

Honoré Hounwanou

Table des matières

1. Introduction

Mama Miya,

Hello guys, heureux de vous retrouver dans cette nouvelle vidéo des TEACHERS DU NET[1] :).

Alors si vous suivez la série de vidéos sur **la création d'un réseau social en PHP**, vous avez sûrement remarqué qu'aucune vidéo n'a été postée ces jours-ci.

Je tiens encore une fois à m'en excuser, cela est indépendant de ma volonté.

Je suis actuellement dans un endroit où il y a quotidiennement du bruit, ce qui m'empêche de faire des vidéos. Bien sûr, je pourrai m'entêter à les faire, mais vous aurez sans doute du mal à m'entendre et de plus la qualité de la vidéo en sera détériorée. A quoi bon alors ?

Ce matin en réfléchissant pendant quelques minutes, j'ai eu à trouver cette solution : *Faire des tutoriels écrits le temps que je finisse ma petite excursion dans ce monde de bruit :).*

Ne vous inquiétez surtout pas, cela est à titre **temporaire**. En effet, dès que je serai de retour à mon appartement, je referai des versions vidéos de ces tutoriels écrits. Le plus important pour moi est de faire en sorte que vous n'abandonniez pas la série.

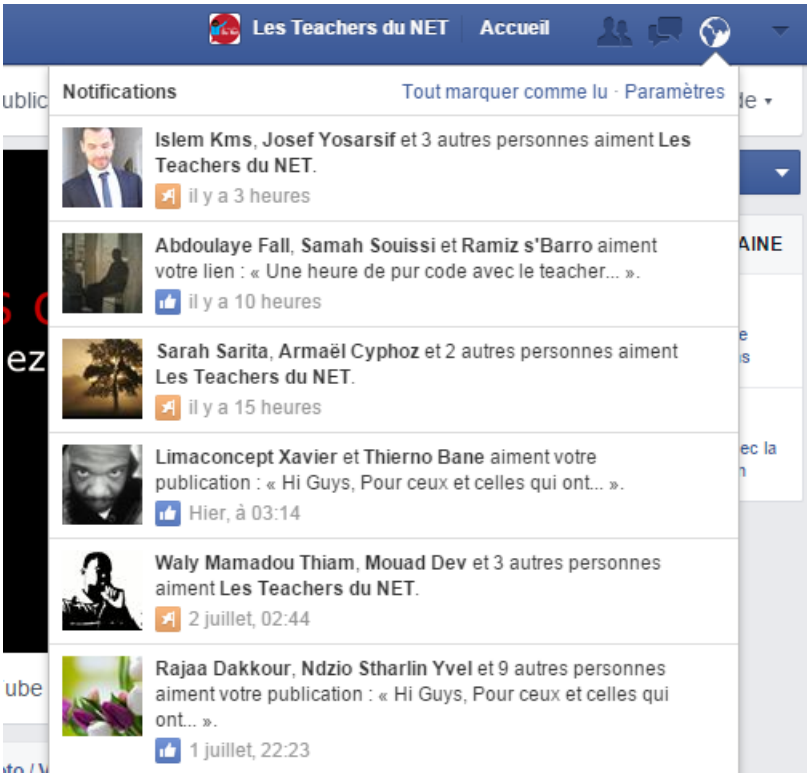
Le décor étant planté, les choses sérieuses peuvent enfin démarrer…

1.1 Qu'allons nous faire dans ce tutoriel ?

Pour le moment, pour savoir si une personne nous a envoyé une demande d'amitié, il nous faut nous rendre manuellement sur la page de profil de cette personne. Il en est de même si nous souhaitons vérifier si un utilisateur a accepté notre demande d'amitié. Vous conviendrai avec moi que cela n'est pas du tout efficient et peut s'avérer très fatiguant.

Nous allons donc dans ce tutoriel, tenter d'automatiser tout cela en mettant en place un petit système de notifications, un peu comme sur Facebook.

[1] http://youtube.com/hounwanou1993

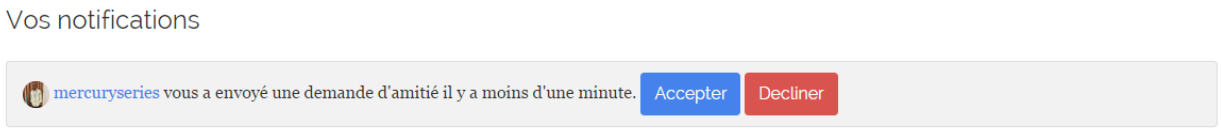


Système de notifications de Facebook

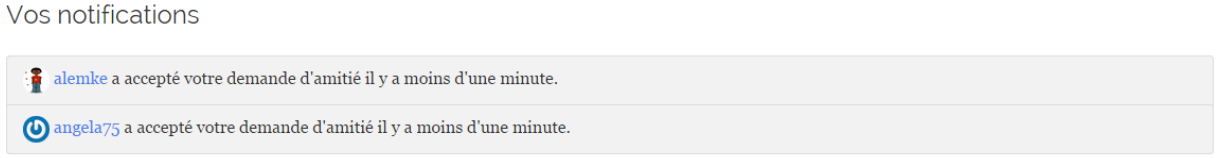
Pour le moment, sur notre mini réseau social, nos utilisateurs seront uniquement notifiés :

- Lorsqu'ils recevront une nouvelle demande d'amitié et
- Lorsqu'on acceptera leur demande d'amitié.

Nous aurons donc des messages de notifications de ce type :



Exemple de message de notification



Exemple de message de notification

Alors ready ? Let’s go !

Attention !

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

2. La table notifications

La première des choses à laquelle nous allons nous intéresser est la structure de notre table `notifications`. Cette table comme vous l'aurez deviné sera chargée de contenir l'ensemble de nos notifications.

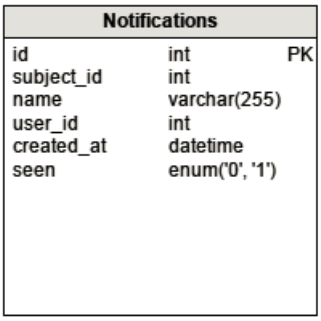
Pour l'instant, nous pouvons poser les hypothèses suivantes :

- Une notification appartient à un et un seul utilisateur.
- Un utilisateur peut avoir une ou plusieurs notifications.

Nous avons donc affaire à une relation *Père - Fils* ou encore *One to Many* pour les amoureux de la langue de Shakespeare.

Nous aurons donc besoin d'une **clé étrangère** `user_id` au niveau de la table `notifications`.

Pour ce faire, je vous propose donc la structure suivante :



La table notifications

et voici la requête SQL qui va avec :

```sql
CREATE TABLE notifications(
  id INT PRIMARY KEY AUTO_INCREMENT,
  subject_id INT(11),
  name VARCHAR(255),
  user_id INT(11),
  created_at DATETIME DEFAULT NOW(),
  seen ENUM('0', '1') DEFAULT '0',
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

- Le champ `id` va représenter la clé primaire de notre table `notifications`. C'est un entier en `AUTO_INCREMENT`. Rien de bien nouveau pour vous !
- Le champ `name` va représenter le nom de la notification. Nous y reviendrons dans le **chapitre 4**.
- Le champ `user_id` va représenter l'identifiant de l'utilisateur déclenchant l'action (par exemple l'identifiant de celui qui envoie la demande d'amitié ou encore l'identifiant de celui qui accepte la demande d'amitié).
- Le champ `subject_id` va représenter l'identifiant du sujet de la notification. En grosso modo, celui qui reçoit l'action (par exemple l'identifiant de celui à qui on a envoyé une demande d'amitié ou encore l'identifiant de celui dont on accepte la demande d'amitié).
- Le champ `created_at` va représenter la date et l'heure à laquelle la notification a été créée.
- Le champ `seen` va permettre de savoir si oui ou non la notification a déjà été vue par l'utilisateur afin de ne pas la lui réafficher plusieurs fois. Si `seen` vaut `0` cela signifiera que la notification n'a pas encore été lue par notre utlisateur. Si `seen` vaut `1` par contre, cela signifiera que l'utilisateur a bel et bien lue la notification.
- L'ajout de `FOREIGN KEY (user_id) REFERENCES users(id)` permet de dire que le champ `user_id` est une clé étrangère qui fait ici référence à la clé primaire `id` de la table `users`. On avait utilisé ce mushibishi lorsqu'on créait la table `friends_relationships`.

Une question que vous pouvez me poser, c'est pourquoi nous n'avons pas mis également `FOREIGN KEY (subject_id) REFERENCES users(id)` étant donné que `subject_id` sera lui aussi l'identifiant d'un utilisateur présent au niveau de la table `users` ?

La réponse est toute simple, `subject_id` ne sera pas forcément l'identifiant d'un utilisateur présent au niveau de la table `users` :).

En effet, il est clair que celui qui déclenchera l'action sera toujours un utilisateur, raison pour laquelle j'ai eu à mettre `user_id` et j'ai de plus rajouté cette contrainte de clé étrangère.

Pour le `subject_id` par contre, il peut aussi bien s'agir de l'identifiant d'un utilisateur ou de l'identifiant d'un commentaire ou encore de l'identifiant d'un message et que sais-je encore. Ainsis si par la suite vous souhaitez avoir des notifications du genre *mercuryseries a aimé votre publication* ou encore *alemke a aimé votre commentaire*, vous n'aurez pas besoin de créer une nouvelle table.

Si vous aviez mis par contre `user_receiver_id` en lieu et place de `subject_id`, il aurait été difficile de comprendre que ce champ pouvait contenir autre chose que l'identifiant d'un utilisateur.

J'espère que les choses sont maintenant claires. Si tel n'est toujours pas votre cas, sentez vous libre de m'envoyer un mail à cette adresse **mercuryseries@gmail.com**, je me ferai un plaisir d'en discuter davatange avec vous.

Parfait !

Il nous faut à présent nous connecter nous à votre serveur MySQL :

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

Ensuite spécifier que nous souhaitons utiliser la base de données `boom` :

```
use boom;
```

Puis créer la table `notifications` à proprement parler en exécutant la requête SQL suivante :

```
CREATE TABLE notifications(
  id INT PRIMARY KEY AUTO_INCREMENT,
  subject_id INT(11),
  name VARCHAR(255),
  user_id INT(11),
  created_at DATETIME DEFAULT NOW(),
  seen ENUM('0', '1') DEFAULT '0',
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Et voilà, notre fameuse table `notifications` a maintenant été créée. C'est à présent le moment de passer à autre chose. N'êtes-vous pas d'accord ?

3. L’indicateur de notifications

Dans ce chapitre qui je crois bien sera relativement court, nous allons nous intéresser à l’ajout d’un indicateur de notifications.

Pour le coup, j’ai voulu faire très simple. Nous allons tout simplement rajouter au niveau de la barre de navigation une icône de cloche à côté de laquelle se trouvera un nombre représentant le nombre de notifications non lues.

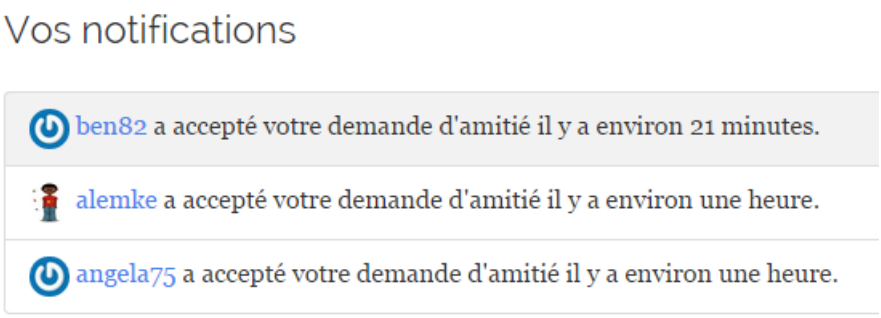
Vu qu’une image vaut mieux que mille mots, je vous présente ce à quoi ressemblera notre indicateur de notifications.

Indicateur de notifications

Lorsqu’on cliquera sur cette icône, cela nous redirigera vers la page `notifications.php` qui sera chargée d’afficher les notifications concernant l’utilisateur connecté.

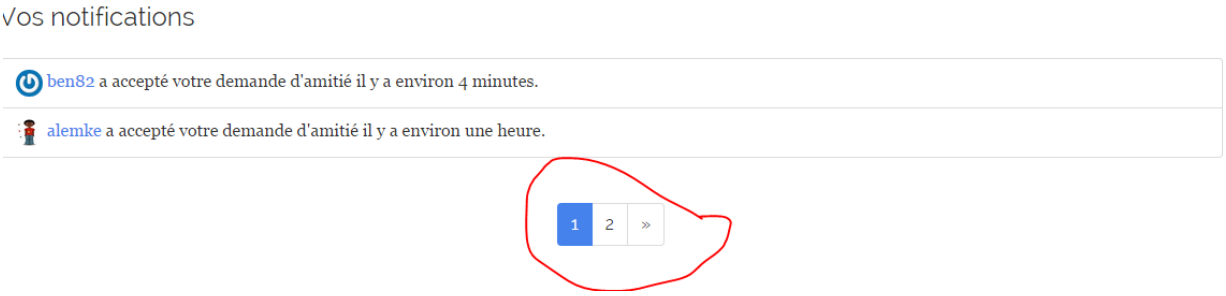
Page des notifications

Les notifications non lues auront un joli petit fond gris. Quant aux notifications déjà lues, elles garderont le fond blanc qui est présent par défaut.



Notifications lues et non lues

Nous donnerons également la possibilité à l'utilisateur de pouvoir avoir une pagination pour l'ensemble de ses notifications.



Pagination pour les notifications

Nous afficherons normalement 10 notifications par page. Mais là histoire de vous montrer ce à quoi ressemblera la pagination, j'ai juste eu à fixer le nombre de notifications par page à 2.

3.1 Ajout du code HTML

Ouvrir sans plus tarder le fichier `partials/_nav.php` afin que nous puissons rajouter le code HTML requis pour l'affichage de l'indicateur de notifications.

Copiez ce code

```
<li>
    <a href="notifications.php"><i class="fa fa-bell"></i> (10)</a>
</li>
```

et collez-le à l'intérieur de `<ul class="nav navbar-nav navbar-right">` comme ceci :

```
<ul class="nav navbar-nav navbar-right">
    <?php if( is_logged_in() ): ?>
        <li class="dropdown">
            ...
        </li>
        <li>
            <a href="notifications.php"><i class="fa fa-bell"></i> (10)</a>
        </li>
    <?php else: ?>
        ...
    <?php endif; ?>
</ul>
```

Information

`<i class="fa fa-bell"></i>` nous permet d'afficher l'icône **cloche** de font awesome[2].

Le contenu final de votre fichier `partials/_nav.php` devrait ressembler à quelque chose de ce genre :

```
<div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="container">
        <div class="navbar-header">
            ...
        </div>
        <div class="collapse navbar-collapse">
            <ul class="nav navbar-nav">
                <li><a href="list_users.php">Liste des utilisateurs</a></li>
            </ul>
            <ul class="nav navbar-nav navbar-right">
              <?php if( is_logged_in() ): ?>
                <li class="dropdown">
                  ...
                </li>
                <li>
                  <a href="notifications.php"><i class="fa fa-bell"></i> (10)</a>
                </li>
              <?php else: ?>
                ...
              <?php endif; ?>
            </ul>
```

[2] http://fortawesome.github.io/Font-Awesome/icon/bell/

```
        </div>
        <!--/.nav-collapse -->
    </div>
</div>
```

Ouvrez ensuite votre navigateur, votre menu devra ressembler à cela :

Aperçu du menu après modification

Afin de styliser un tant soit peu notre indicateur de notifications, nous allons lui rajouter une classe `have_notifs`. Si notre élément de liste a cette classe `have_notifs`, cela signifiera qu'il y a des notifications disponibles (en d'autres termes que le nombre de notifications est supérieur à 0). Ainsi, pour que l'utilisateur puisse clairement le voir, nous allons changer la couleur du lien se trouvant à l'intérieur de élément de liste. J'ai choisi ici d'utiliser la couleur rouge, mais vous êtes tout à fait libre d'utiliser la couleur qui vous convient.

Remplacez donc

```
<li>
    <a href="notifications.php"><i class="fa fa-bell"></i> (10)</a>
</li>
```

par

```
<li class="have_notifs">
    <a href="notifications.php"><i class="fa fa-bell"></i> (10)</a>
</li>
```

3.2 Ajout du code CSS

Ouvrez ensuite votre fichier `assets/css/main.css` et rajouter ce code :

```
li.have_notifs > a{
    color: red !important;
    font-weight: bold;
}
```

Cela permet de dire que tous liens qui seront des enfants directs d'un élément de liste ayant la classe `have_notifs` se verront affecter la couleur rouge et seront mis également en gras.

Ouvrez à nouveau votre navigateur, votre menu devra ressembler à présent à ceci :

Aperçu du menu après ajout de la classe have_notifs

Parfait !

C'est bien beau tout cela, mais pour l'instant nous avons fixé la valeur **10** pour le nombre de notifications non lues. La classe `have_notifs` a été également ajoutée manuellement. Tentons donc d'ajouter un peu de dynamisme à tout cela en utilisant le langage PHP sinon notre utilisateur se plaindra de tout le temps avoir **10** notifications fictives. Vous risquez d'aller en prison :).

4. Sauvegarde des notifications

Pour afficher des notifications, il va falloir les créer. Eh bien, c'est justement ce que nous allons faire dans ce chapitre. Comme vous allez le voir, ce sera relativement simple.

4.1 Ajout d'une notification lors de l'envoi d'une demande d'amitié.

Ouvrez le fichier `add_friend.php` et rajouter ce code

```
$q = $db->prepare('INSERT INTO notifications(subject_id, name, user_id)
                   VALUES(:subject_id, :name, :user_id)');
$q->execute([
    'subject_id' => $id,
    'name' => 'friend_request_sent',
    'user_id' => get_session('user_id'),
]);
```

Vous devez le mettre tout juste après :

```
$q = $db->prepare('INSERT INTO friends_relationships(user_id1, user_id2)
                   VALUES(:user_id1, :user_id2)');
$q->execute([
    'user_id1' => get_session('user_id'),
    'user_id2' => $id
]);
```

Au final, nous devons donc avoir quelque chose qui ressemble à ceci :

```php
<?php
session_start();

require("includes/init.php");
include('filters/auth_filter.php');

if(!empty($_GET['id']) && $_GET['id'] !== get_session('user_id')){
        if(!if_a_friend_request_has_already_been_sent(get_session('user_id'), $_GET['id\
']')){
        $id = $_GET['id'];

        $q = $db->prepare('INSERT INTO friends_relationships(user_id1, user_id2)
                           VALUES(:user_id1, :user_id2)');
        $q->execute([
            'user_id1' => get_session('user_id'),
            'user_id2' => $id
        ]);


        // Sauvegarde de la notification
        $q = $db->prepare('INSERT INTO notifications(subject_id, name, user_id)
                           VALUES(:subject_id, :name, :user_id)');
        $q->execute([
            'subject_id' => $id,
            'name' => 'friend_request_sent',
            'user_id' => get_session('user_id'),
        ]);

        set_flash("Votre demande d'amitié a été envoyée avec succès!");
        redirect('profile.php?id='.$id);
    } else {
        set_flash("Cet utilisateur vous a déja envoyé une demande d'amitié.");
        redirect('profile.php?id='.$_GET['id']);
    }
} else {
    redirect('profile.php?id='.get_session('user_id'));
}
```

Expliquons à présent le code que nous avons rajouté même si je sais très bien qu'il est assez facile à comprendre.

```php
$q = $db->prepare('INSERT INTO notifications(subject_id, name, user_id)
                   VALUES(:subject_id, :name, :user_id)');
$q->execute([
    'subject_id' => $id,
    'name' => 'friend_request_sent',
    'user_id' => get_session('user_id'),
]);
```

Nous faisons un nouvel enregistrement au niveau de la table `notifications`. Nous indiquons que :

- Le `subject_id` correspond à `$id` qui n'est rien d'autre que `$_GET['id']`. Rappelez vous `$_GET['id']` représente l'identifiant de celui à qui on envoie la demande d'amitié. C'est donc le **receveur de l'action**. Parfait !
- Pour le `name`, nous avons mis `'friend_request_sent'`. Ce nom nous permettra plus tard de savoir quel type de message il faudra afficher. Avec comme valeur `'friend_request_sent'`, nous saurons que cette notification a rapport avec l'envoi d'une nouvelle demande d'amitié. Si cela vous semble confus, ne vous inquiétez pas, vous verrez son utilité par la suite.
- Pour le `user_id`, nous avons mis `get_session('user_id')` qui n'est rien d'autre que l'identifiant de l'utilisateur connecté. Qui déclenche l'action ou devrais-je dire qui envoie la demande d'amitié ? C'est l'utilisateur connecté :).

Voilà donc comment on crée une nouvelle notification. Assez simple n'est-ce pas ?

4.2 Ajout d'une notification lors de l'acceptation d'une demande d'amitié.

Maintenant que vous savez comment crée une nouvelle notification, tout devient simple.

Ouvrez le fichier `acccept_friend_request.php` et rajouter ce code :

```php
$q = $db->prepare('INSERT INTO notifications(subject_id, name, user_id)
                   VALUES(:subject_id, :name, :user_id)');
$q->execute([
    'subject_id' => $id,
    'name' => 'friend_request_accepted',
    'user_id' => get_session('user_id'),
]);
```

Vous devez le mettre tout juste après :

```
$q = $db->prepare("UPDATE friends_relationships
                   SET status = '1'
                   WHERE (user_id1 = :user_id1 AND user_id2 = :user_id2)
                   OR (user_id1 = :user_id2 AND user_id2 = :user_id1)");
$q->execute([
    'user_id1' => get_session('user_id'),
    'user_id2' => $id
]);
```

Au final, nous devons donc avoir quelque chose qui ressemble à ceci :

```
<?php
session_start();

require("includes/init.php");
include('filters/auth_filter.php');


if(!empty($_GET['id']) && $_GET['id'] !== get_session('user_id')){
    $id = $_GET['id'];

    $q = $db->prepare("UPDATE friends_relationships
                       SET status = '1'
                       WHERE (user_id1 = :user_id1 AND user_id2 = :user_id2)
                       OR (user_id1 = :user_id2 AND user_id2 = :user_id1)");
    $q->execute([
        'user_id1' => get_session('user_id'),
        'user_id2' => $id
    ]);


    // Sauvegarde de la notification
    $q = $db->prepare('INSERT INTO notifications(subject_id, name, user_id)
                       VALUES(:subject_id, :name, :user_id)');
    $q->execute([
        'subject_id' => $id,
        'name' => 'friend_request_accepted',
        'user_id' => get_session('user_id'),
    ]);

    set_flash("Vous êtes à présent ami avec cet utilisateur!");
    redirect('profile.php?id='.$id);
```

```
32 } else {
33     redirect('profile.php?id='.get_session('user_id'));
34 }
```

Je suis convaincu que sans explication, vous êtes à même de pouvoir comprendre le bout de code que nous avons rajouté. Mais histoire d'être sûr que nous sommes sur la même longueur d'ondes, je tiens tout de même à l'expliquer. Et comme on le dit très souvent, la répétition est pédagogique :).

```
1 $q = $db->prepare('INSERT INTO notifications(subject_id, name, user_id)
2                    VALUES(:subject_id, :name, :user_id)');
3 $q->execute([
4     'subject_id' => $id,
5     'name' => 'friend_request_accepted',
6     'user_id' => get_session('user_id'),
7 ]);
```

Nous faisons un nouvel enregistrement au niveau de la table `notifications`. Nous indiquons que :

- Le `subject_id` correspond à `$id` qui n'est rien d'autre que `$_GET['id']`. Rappelez vous `$_GET['id']` dans ce cas, représente l'identifiant de celui dont on accepte la demande d'amitié. C'est donc le **receveur de l'action**. Parfait !
- Pour le `name`, nous avons mis `'friend_request_accepted'`. Ce nom nous permettra plus tard de savoir quel type de message il faudra afficher. Avec comme valeur `'friend_request_accepted'`, nous saurons que cette notification a rapport avec l'acceptation d'une demande d'amitié. Comme vous pouvez le voir, c'est la seule chose qui change par rapport à l'enregistrement précédent.
- Pour le `user_id`, nous avons mis `get_session('user_id')` qui n'est rien d'autre que l'identifiant de l'utilisateur connecté. Qui déclenche l'action ou devrais-je dire qui accepte la demande d'amitié ? C'est l'utilisateur connecté :).

Parfait !

4.3 Test

- Essayer à présent d'envoyer quelques demandes d'amitié, vous verrez qu'en plus de la création d'une nouvelle entrée au niveau de la table `friends_relationships`, une nouvelle notification sera également créée avec les champs `subject_id`, `name`, `user_id` `created_at` et `seen` correctement remplis. A noter que les champs `created_at` et `seen` sont remplis automatiquement étant donné qu'ils possèdent des valeurs par défaut.

Vous pouvez confirmer mes dires en exécutant les requêtes SQL suivantes :

```sql
SELECT * FROM friends_relationships;
SELECT * FROM notifications;
```

- Essayer également d'accepter les demandes d'amitié précédemment envoyées, vous verrez que de nouvelles notifications seront également rajoutées.

Formidable !

Maintenant que nous avons des notifications, ne pensez-vous pas qu'il est grand temps de les afficher :) ?

5. Affichage des notifications

Plus qu'un dernier chapitre avant de boucler notre système de notifications :).

À ce stade, je peux vous dire qu'il est beaucoup plus facile de faire des tutoriels vidéos que d'écrire, car là je suis complètement claqué !

Afin d'être encore motivé à terminer ce tutoriel, je pense pendant quelques minutes au sourire sur votre visage lorsque vous serez en train de lire ces lignes :).

Mika mon chat xD :)

5.1 Rendre dynamique l'indicateur de notifications

Dans un premier temps, nous allons rendre dynamique notre indicateur de notifications.

5.1.1 Nombre de notifications non lues

Pour ce faire, il va nous falloir être en mesure de compter le nombre de notifications non lues pour l'utilisateur présentement connecté.

La requête est vraiment très simple. Je suis convaincu que vous êtes en mesure de l'écrire les yeux fermés.

```
1 SELECT id FROM notifications WHERE subject_id = ? AND seen = '0'
```

Avec du code PHP, cela donne :

```
1 $q = $db->prepare("SELECT id FROM notifications
2                    WHERE subject_id = ? AND seen = '0'");
3
4 $q->execute([get_session('user_id')]);
5
6 $notifications_count = $q->rowCount();
```

Nous sélectionnons ici toutes les notifications non lues dont l'utilisateur connecté en est le receveur.

La variable `$notifications_count` va donc contenir le nombre de notifications non lues pour l'utilisateur actuellement connecté.

La question à se poser maintenant est où mettre ce code ?

Etant donné que l'indicateur de notifications sera affiché sur toutes les pages, il va nous falloir mettre ce code au niveau d'un fichier qui est appelé par toutes les pages de notre site web. Vous voyez de quel fichier je veux parler ?

Eh oui, il s'agit de notre fameux fichier **includes/init.php**.

Ouvrez donc le fichier `includes/init.php` et rajouter ce code :

```
1 $q = $db->prepare("SELECT id FROM notifications
2                    WHERE subject_id = ? AND seen = '0'");
3
4 $q->execute([get_session('user_id')]);
5
6 $notifications_count = $q->rowCount();
```

Voici donc au final le contenu du fichier `includes/init.php` :

```php
<?php
require("bootstrap/locale.php");
require("config/database.php");
require("includes/functions.php");
require("includes/constants.php");

if(!empty($_COOKIE['pseudo']) && !empty($_COOKIE['user_id'])){
    $_SESSION['pseudo'] = $_COOKIE['pseudo'];
    $_SESSION['user_id'] = $_COOKIE['user_id'];
    $_SESSION['avatar'] = $_COOKIE['avatar'];
}

//Récupération du nombre total de notifications non lues
$q = $db->prepare("SELECT id FROM notifications
                   WHERE subject_id = ? AND seen = '0'");

$q->execute([get_session('user_id')]);

$notifications_count = $q->rowCount();

auto_login();
```

5.1.2 Afficher le nombre de notifications non lues

Pour afficher le nombre de notifications non lues, ouvrez le fichier `partials/_nav.php` et remplacer :

```php
<li class="have_notifs">
    <a href="notifications.php"><i class="fa fa-bell"></i> (10)</a>
</li>
```

par :

```php
<li class="<?= $notifications_count > 0 ? 'have_notifs' : '' ?>">
    <a href="notifications.php"><i class="fa fa-bell"></i>
        <?= $notifications_count > 0 ? "($notifications_count)" : ''; ?>
    </a>
</li>
```

- Si le nombre de notifications est supérieur à 0 alors dans ce cas nous ajoutons la classe `have_notifs`, ce qui va permettre d'alerter visuellement l'utilisateur. Dans le cas contraire nous n'attribuons aucune classe à notre élément de liste.
- De même, si le nombre de notifications est supérieur à 0, nous affichons à côté de l'icône cloche, le nombre total de notifications non lues récupéré précédemment. Dans le cas contraire nous n'affichons rien. Juste l'icône cloche.

5.2 La page notifications.php

Nous allons pour terminer nous intéresser à notre fameuse page `notifications.php`. Comme vous allez le voir, nous n'aurons pas de liste déroulante pour afficher les notifications comme sur Facebook. Dans notre cas, pour question de simplicité, les notifications seront affichées sur une page dédiée. Facebook dispose également de cette page notifications[3].

Si vous souhaitez plus tard ajouter également une liste déroulante pour vos notifications, je tiens à informer que vous avez déjà toutes les connaissances requises pour le faire. Ce sera donc votre petit challenge pour ce tutoriel :).

5.2.1 Création des fichiers nécessaires.

Nous allons avoir besoin de créer deux nouveaux fichiers :

- Le fichier `notifications.php` qui va contenir la logique.
- Le fichier `views/notifications.view.php` qui se chargera de simplement afficher les notifications.

Nous aurons également besoin de créer d'autres fichiers, mais nous le ferons le moment opportun.

Le fichier notifications.php

Ouvrez le fichier `notifications.php` et rajoutez le code suivant :

```
1  <?php
2  session_start();
3  require("includes/init.php");
4  include('filters/auth_filter.php');
5
6  // Nous récupérons toutes les notifications de l'utilisateur connecté
7  // (Aussi bien les notifications lues que les notifications non lues).
8  $q = $db->prepare(
9          "SELECT users.pseudo, users.avatar, users.email,
10                 notifications.subject_id,
11                 notifications.name, notifications.user_id,
12                 notifications.seen, notifications.created_at
13         FROM notifications
14         LEFT JOIN users ON users.id = user_id
15         WHERE subject_id = ?
```

[3] https://www.facebook.com/notifications

```
16          ORDER BY notifications.created_at DESC
17      ");
18
19  $q->execute([get_session('user_id')]);
20
21  // Nous les stockons au niveau de la variable $notifications
22  $notifications = $q->fetchAll(PDO::FETCH_OBJ);
23
24  // Après avoir récupéré les notifications de l'utilisateur connecté,
25  // nous modifions la valeur de leur attribut 'seen' afin d'indiquer que
26  // l'utilisateur vient de lire ces notifications.
27  $q = $db->prepare("UPDATE notifications SET seen = '1' WHERE subject_id = ?");
28  $q->execute([get_session('user_id')]);
29
30
31  // Nous affichons ensuite le contenu de notre fichier notifications.view.php
32  require("views/notifications.view.php");
```

Je crois bien que les commentaires inscrits dans le code vous permettront de le comprendre aisément. Néanmoins, j'aimerais revenir sur la requête SQL que nous avons utilisée.

Si on devait la traduire littéralement, c'est comme si on disait :

Sélectionne moi le pseudo, l'avatar et l'email de l'utilisateur connecté de même que les champs subject_id, name, user_id, seen et created_at des notifications dont il en est le receveur tout en les ordonnant par ordre chronologique décroissant. En d'autres termes de la notification la plus récente à la notification la plus ancienne.

5.2.3 Le fichier notifications.view.php

Au niveau du fichier `views/notifications.view.php`, nous avons accès à la variable `$notifications`. Nous pouvons donc utiliser cette dernière afin d'afficher l'ensemble des notifications comme suit :

```
1  <?php $title = "Notifications"; ?>
2  <?php include('partials/_header.php'); ?>
3
4  <div id="main-content">
5      <div class="container">
6          <h1 class="lead">Vos notifications</h1>
7
8          <?php if(count($notifications) > 0): ?>
9            <ul class="list-group">
```

```
                <?php foreach($notifications as $notification): ?>
                    <li class="list-group-item
                        <?= $notification->seen == '0' ? 'not_seen' : '' ?>"
                >
                        <?php require("partials/notifications/{$notification->name}.php"); ?>
                    </li>
                <?php endforeach; ?>
            </ul>
        <?php else: ?>
            <p>Aucune notification disponible pour l'instant.</p>
        <?php endif; ?>
    </div>
</div>

<?php include('partials/_footer.php'); ?>
```

Nous parcourons l'ensemble des notifications, et en fonction du nom de la notification nous incluons le fichier approprié. Comme vous pouvez le voir cela nous permet de mieux structurer notre code source en lieu et place d'avoir faire à des `if - elseif - else` sur le nom de la notification.

De même, si la notification n'a pas encore été lue, nous rajoutons la classe `not_seen` à notre élément de liste, ce qui nous permettra de lui ajouter le joli petit fond gris.

Il nous donc à présent créer la classe `not_seen` et créer les fichiers `partials/notifications/-friend_request_sent.php` et `partials/notifications/friend_request_accepted.php` vu que nos deux noms de notifications actuels sont respectivement `friend_request_sent` et `friend_request_-accepted`.

Le fichier partials/notifications/friend_request_sent.php

Ouvrez le fichier `partials/notifications/friend_request_sent.php` et rajouter le code suivant :

```
<a href="profile.php?id=<?= $notification->user_id ?>">
    <img src="<?= $notification->avatar ? $notification->avatar : get_avatar_url\
($notification->email, 40) ?>" alt="Image de profil de <?= e($notification->pseu\
do) ?>" class="avatar-xs">
    <?= e($notification->pseudo) ?>
</a>
vous a envoyé une demande d'amitié <span class="timeago" title="<?= $notificatio\
n->created_at ?>"><?= $notification->created_at ?></span>.
<a class="btn btn-primary" href="accept_friend_request.php?id=<?= $notification-\
>user_id ?>">Accepter</a>
<a class="btn btn-danger" href="delete_friend.php?id=<?= $notification->user_id \
?>">Decliner</a>
```

Le fichier partials/notifications/friend_request_accepted.php

Ouvrez le fichier `friend_request_accepted.php` et rajouter le code suivant :

```
<a href="profile.php?id=<?= $notification->user_id ?>">
    <img src="<?= $notification->avatar ? $notification->avatar : get_avatar_url\
($notification->email, 40) ?>" alt="Image de profil de <?= e($notification->pseu\
do) ?>" class="avatar-xs">
    <?= e($notification->pseudo) ?>
</a>
a accepté votre demande d'amitié <span class="timeago" title="<?= $notification-\
>created_at ?>"><?= $notification->created_at ?></span>.
```

Rajouter la classe not_seen

Ouvrez le fichier `assets/css/main.css` et rajouter le code suivant :

```
.not_seen{
    background-color: #f2f2f2;
}
```

Inclusion de la librairie timeago

Comme vous l'avez vu, nous avons eu à utiliser la librairie `timeago`. Il va donc nous falloir la charger.

Ouvrez donc le fichier `views/notifications.view.php` et remplacer :

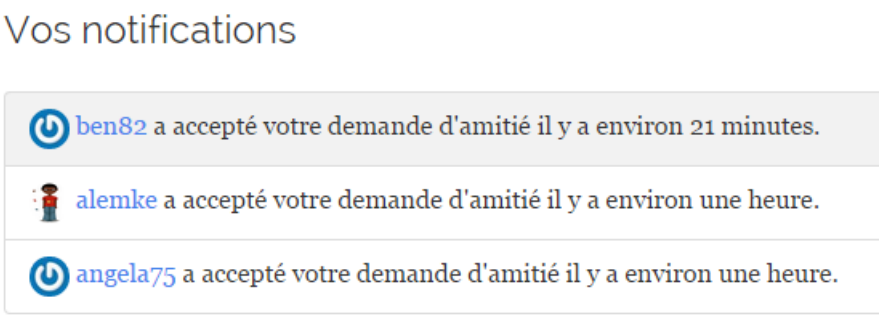
```
<?php include('partials/_footer.php'); ?>
```

par :

```
<!-- SCRIPTS -->
<script src="assets/js/jquery.min.js"></script>
<script src="assets/js/main.js"></script>
<script src="assets/js/bootstrap.min.js"></script>
<script src="assets/js/jquery.timeago.js"></script>
<script src="assets/js/jquery.timeago.fr.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
      $(".timeago").timeago();
    });
</script>
</body>
</html>
```

5.2.4 Test

Ouvrez à présent votre navigateur et assurez vous que le système de notifications fonctionne parfaitement.



Système de notifications fonctionnel

5.2.5 Ajout de la pagination

Le code de pagination est assez similaire au code que nous avons utilisé au niveau du fichier `list_users.php`. Je ne vais donc pas l'expliquer. Je vais me contenter de vous fournir les nouveaux contenus des fichiers `notifications.php` et `views/notifications.php`.

Nouveau contenu de notifications.php

```
1  <?php
2  session_start();
3  require("includes/init.php");
4  include('filters/auth_filter.php');
5
6
7  $q = $db->prepare(
8          "SELECT notifications.id FROM notifications
9          LEFT JOIN users ON users.id = user_id
10         WHERE subject_id = ?
11     ");
12 $q->execute([get_session('user_id')]);
13
14 $notifications_total = $q->rowCount();
15
16
17 if($notifications_total >= 1){
```

```
18
19      $nbre_notifications_par_page = 10;
20
21      $nbre_pages_max_gauche_et_droite = 4;
22
23      $last_page = ceil($notifications_total / $nbre_notifications_par_page);
24
25      if(isset($_GET['page']) && is_numeric($_GET['page'])){
26          $page_num = $_GET['page'];
27      } else {
28          $page_num = 1;
29      }
30
31      if($page_num < 1){
32          $page_num = 1;
33      } else if($page_num > $last_page) {
34          $page_num = $last_page;
35      }
36
37      $limit = 'LIMIT '.($page_num - 1) * $nbre_notifications_par_page. ',' . $nbr\
38  e_notifications_par_page;
39
40
41      $q = $db->prepare(
42              "SELECT users.pseudo, users.avatar, users.email,
43                      notifications.subject_id, notifications.name,
44                      notifications.user_id, notifications.seen,
45                      notifications.created_at
46              FROM notifications
47              LEFT JOIN users ON users.id = user_id
48              WHERE subject_id = ?
49              ORDER BY notifications.created_at DESC
50              $limit
51          ");
52      $q->execute([get_session('user_id')]);
53
54      $notifications = $q->fetchAll(PDO::FETCH_OBJ);
55
56
57      $pagination = '<nav class="text-center"><ul class="pagination">';
58
59      if($last_page != 1){
```

```
60          if($page_num > 1){
61              $previous = $page_num - 1;
62              $pagination .= '<li><a href="notifications.php?page='.$previous.'" a\
63  ria-label="Precedent"><span aria-hidden="true">&laquo;</span></a></li>';
64
65              for($i = $page_num - $nbre_pages_max_gauche_et_droite; $i < $page_nu\
66  m; $i++){
67                  if($i > 0){
68                      $pagination .= '<li><a href="notifications.php?page='.$i.'">\
69  '.$i.'</a></li>';
70                  }
71              }
72          }
73
74          $pagination .= '<li class="active"><a href="#">'.$page_num.'</a></li>';
75
76          for($i = $page_num+1; $i <= $last_page; $i++){
77              $pagination .= '<li><a href="notifications.php?page='.$i.'">'.$i.'</\
78  a></li>';
79
80              if($i >= $page_num + $nbre_pages_max_gauche_et_droite){
81                  break;
82              }
83          }
84
85         if($page_num != $last_page){
86              $next = $page_num + 1;
87              $pagination .= '<li><a href="notifications.php?page='.$next.'"aria-l\
88  abel="Suivant"><span aria-hidden="true">&raquo;</span></a></li>';
89          }
90      }
91
92      $pagination .= '</ul></nav>';
93
94      require("views/notifications.view.php");
95  } else {
96      set_flash('Aucune notification disponible pour le moment...');
97      redirect('index.php');
98  }
99
100 $q = $db->prepare("UPDATE notifications SET seen = '1' WHERE subject_id = ?");
101 $q->execute([get_session('user_id')]);
```

Nouveau contenu de views/notifications.view.php

```
<?php $title = "Notifications"; ?>
<?php include('partials/_header.php'); ?>

<div id="main-content">
    <div class="container">
        <h1 class="lead">Vos notifications</h1>

        <ul class="list-group">
                    <?php foreach($notifications as $notification): ?>
            <li class="list-group-item
                        <?= $notification->seen == '0' ? 'not_seen' : '' ?>"
            >
                <?php require("partials/notifications/{$notification->name}.php"\
); ?>
            </li>
            <?php endforeach; ?>
        </ul>

        <div id="pagination"><?= $pagination ?></div>
    </div>
</div>

<?php include('partials/_footer.php'); ?>
```

Conclusion

Houra ! Nous avons maintenant un système de notifications qui fonctionne. Comme vous avez pu le voir il n'y avait vraiment rien de bien compliqué. Il reste encore des choses à améliorer mais pour cela je vous fais confiance. Vous pourrez vous servir de ce qu'on eu à faire dans ce tutoriel comme base et l'améliorer en conséquence.

D'ici là portez vous bien, et je crois bien qu'il est maintenant l'heure pour moi de regarder une bonne série.

Avez-vous des propositions de séries comiques ? je suis preneur :) **mercuryseries@gmail.com**.