Université Toulouse III – Paul sabatier
L2 Informatique

Antoine de Roquemaurel (antoine.de-roquemaurel@univ-tlse3.fr)
Groupe 2.2

Conception d'une application de folksonomies

Systèmes d'information et applications Web

Toulouse, le 3 mai 2013

Avant-propos

Ce dossier comporte les différentes de réalisation d'une application Web de folksonomies, site permettant de partager des URL et de les associer à des tages. Il à été conçut par Antoine de Roquemaurel dans le cadre du module *Systèmes d'Information et Application Web* de la L2 Informatique de l'université Toulouse III – Paul Sabatier.

Tester le projet

L'archive que vous avez reçus est organisée comme ceci :

folksonomies/ Contient tous les fichiers du Site Web.

rapport.pdf Le présent rapport que vous êtes en train de lire

Afin de tester le projet, vous pouvez avoir accès au site web fonctionnel directement en ligne à l'adresse `http://dev.joohoo.fr/dev/folksonomies/`. Il est également possible d'utiliser notre code source avec votre propre serveur web et votre base de données, pour cela les paramétrage des accès à la base de données sont présent dans le fichier `folksonomies/database/connect.php`. Toutes les adresses web étant en relatifs, aucun problème ne devrait avoir lieu.

Ce site Web utilisant des fonctionnalités de HTML5 et CSS3, il est recommandé d'utiliser un navigateur récent. Ce site à été développé sous Google Chrome, ainsi l'affichage sera optimal sur ce navigateur, cependant il devrait s'afficher correctement sur les autres.

Il est possible de créer le schéma de la base de données grâce au menu Base de données → Création de la base.

Rédigé le 3 mai 2013 par Antoine de Roquemaurel.

Table des matières

1 Les besoins de l'application **4**

2 Organisation du travail **6**

2.1 Un outil de gestion de projet : Redmine 6

2.2 Un logiciel de versionnement : Git 7

3 Implémentation et conception **8**

3.1 Mise en forme : les vues 8

3.1.1 Le CSS 8

3.1.2 Le JavaScript 9

3.1.3 Le HTML 9

3.2 Connexions à la base de données : Les modèles 10

3.3 Liaison des vues aux modèles : Les contrôleurs 11

4 Résultats obtenus **12**

4.1 Nuage de tags et recherche 12

4.2 Liste de tous les documents 13

4.3 Liste des documents d'un utilisateur 13

4.4 Affichage de « Mes documents » 13

4.5 Création et suppression de la base de données 14

A Table des figures **15**

B Liste des codes sources **15**

Chapitre 1

Les besoins de l'application

Ce projet consiste en la création d'un site web utilisant les technologies HTML, CSS, PHP et la base de données MySQL permettant de gérer des tags sur des sites web. Ces tags permettent de retrouver facilement des URL, d'effectuer des recherches en fonction de tags, ...

Pour cela, un modèle de base de données nous à été fournis, mon application ayant été légèrement améliorée afin de pouvoir gérer une multitude d'utilisateurs, celui-ci à été légèrement modifié. Figure 2.1 est présent le nouveau modèle.

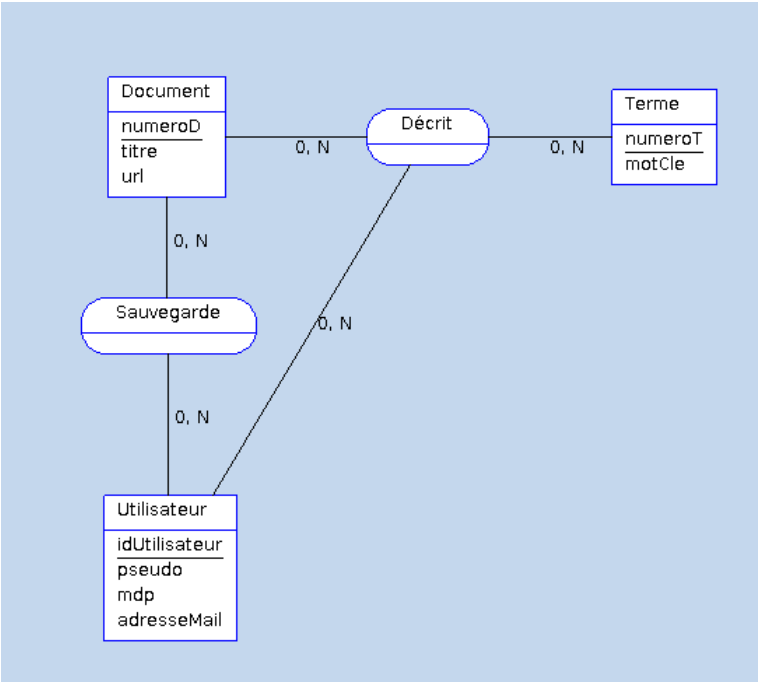

Figure 1.1 – Modèle conceptuel de données de l'application

Insertion d'un nouveau document Le site doit permettre d'ajouter un nouveau document, et de le lier à des tags. Si l'adresse du site existe déjà dans la base de données, celle-ci n'est pas ajoutée de nouveau, seuls les nouveaux tags sont ajoutés.

Lister les documents enregistrés sur le site Il doit être possible de lister tous les documents présent dans la base de données, ainsi que les tags associés. Il est également possible de savoir quels utilisateur ont enregistrés un document.

Afficher un nuage de tag L'application doit pouvoir afficher un « tag cloud » ou nuage de tag. Les différents tag s'affiche d'une taille différente et d'une couleur plus foncée en fonctions de leur importance, c'est-à-dire du nombre de document liés à ce tag.

Afficher tous les docuemnts liés à un tag donné Il doit être possible d'afficher tous les documents étant liés à un tag donné.

Rechercher un document On peut rechercher un document en fonction d'un ou plusieurs tags. Si plusieurs tags sont rensignés, la rechercher retournera tous les documents étant associés à au moins un des tags.

Inscription d'un nouvel utilisateur Il est également possible de s'inscrire afin d'utiliser le système, pour cela un pseudo, une adresse e-mail et un mot de passe sont demandés, les mots de passes sont cryptés dans la

base de données. L'inscription d'un utilisateur lui permet ensuite d'ajouter de nouveaux documents, de leur donner une description personnalisée et de lister tous les documents lui appartenant.

Afficher les documents enregistrés par un utilisateur Enfin, un utilisateur peut afficher uniquement les documents lui appartenant.

Chapitre 2

Organisation du travail

Pour ce projet, j'étais seul, ainsi cela fut plus simples que lors du projet précédent où nous étions deux à travailler dessus.

Cependant, afin de bien organiser mon travail, de n'oublier aucune fonctionnalités et de ne rien perdre en cas de problème technique, j'ai choisis pour ce projet d'utiliser un outil de gestion de projet et un logiciel de versionnement de la même manière que le projet précédent.

2.1 Un outil de gestion de projet : Redmine

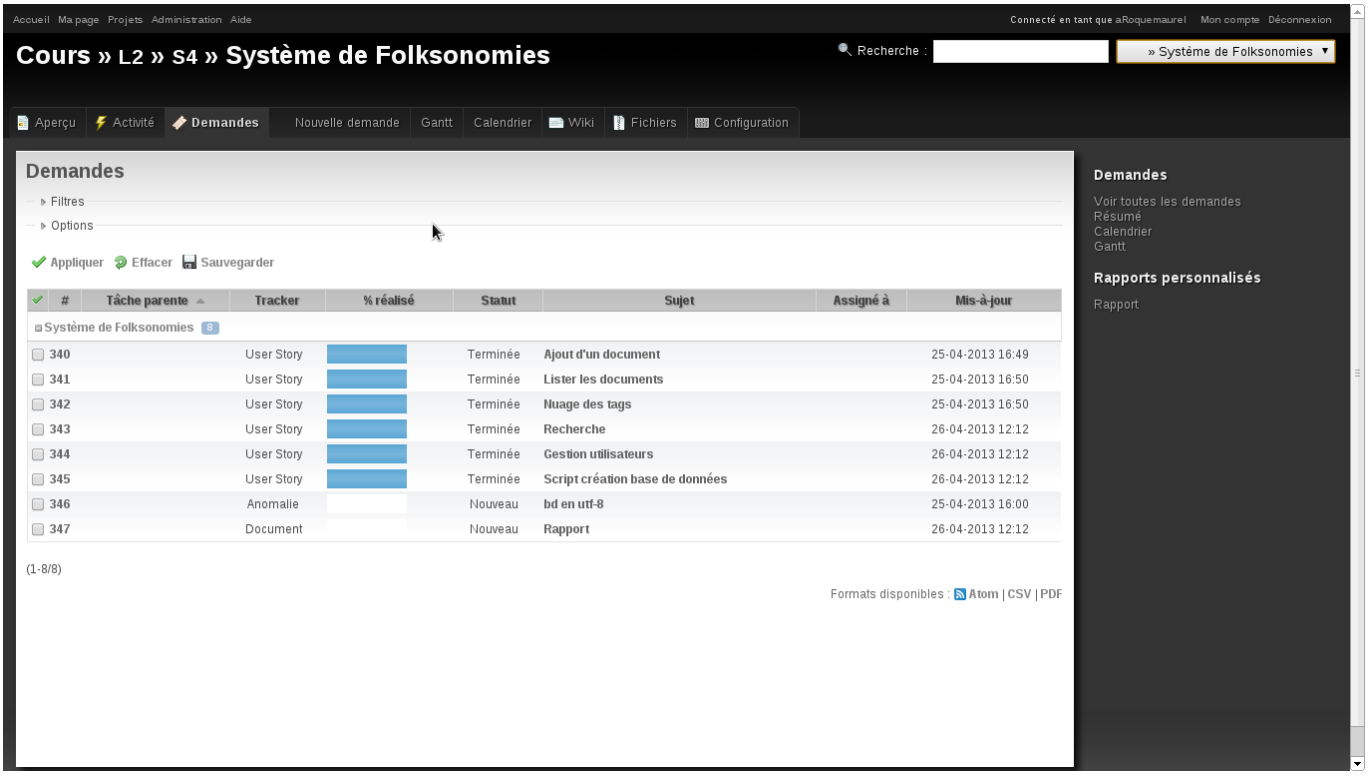

Figure 2.1 – Affichage des demandes dans Redmine

Pour le projet, j'ai utilisé *Redmine*, une plateforme web de gestion de projet (Cf figure 2.1). Elle m'a permis de simplifier le travail, et de ne rien oublier.

En effet, je pouvais créer des tâches, signaler qu'elles sont en cours/terminés/en tests, leur donner des dates limites, etc...

2.2 Un logiciel de versionnement : Git

Afin de limiter les problèmes liés à une perte de données, j'ai utilisé un logiciel de versionnement Git. Il a deux intérêt, tout d'abord, je pouvais travailler de plusieurs ordinateurs en parallèle sur le projet sans me soucier de fusionner le travail [1].

D'autre part, tous les logs étant enregistrés, je pouvais voir l'avancée du projet.

Enfin, toutes les modification sont stockées sur le serveur, ainsi en cas de problème, il est très facile de revenir à la version précédente ou même de comparer deux versions afin de voir les changements et de comprendre rapidement pourquoi une fonctionnalité à régressé.

1. À condition de ne pas travailler sur deux lignes de code identiques

Chapitre 3

Implémentation et conception

La conception du projet est l'étape la plus importante, ainsi j'ai préféré repenser à ma manière le projet, je suis partis sur une architecture respectant le patron de conception MVC [1], c'est à dire que toute la partie affichage (notamment le HTML), Modèle (c'est-à-dire les requêtes SQL) sont séparés, c'est le Contrôleur qui dirige le modèle et la vue, c'est ainsi le *chef d'orchestre*.

3.1 Mise en forme : les vues

Afin d'avoir un site web élégant sans passer beaucoup de temps à réinventer la roue en CSS, j'ai choisi d'utiliser un framework [2] CSS appelé *bootstrap*. Celui-ci nous permet d'avoir une mise en forme sobre très rapidement. La mise en forme est organisé comme suit :

3.1.1 Le CSS

Le CSS est présent dans le dossier `style/css`, dedans est présent d'une part le CSS fourni par *bootstrap*, auquel je n'ai pas touché, d'autre part, mon CSS qui surcharge certains affichage de *bootstrap*, celui-ci est dans `style/css/style.css`.

```
/* Affichage du nuage de tag */
#cloudWords {
  width: 90%;
  text-align: center;
  margin: auto;
}
.keyword {
  display: inline-block;
  margin-top: 30px;
}

/* Champ de recherche */
#searchField {
  margin-top:10px;
  margin-bottom: -10px;
}

#search {
  margin-top: 50px;
  margin-left: 15px;
}
```

Listing 3.1 – Code CSS

1. Modèles Vue Contrôleur ou *pattern MVC* : Model View Controller en anglais
2. Kit de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture). — D'après *wikipédia*

3.1.2 Le JavaScript

Le JavaScript quant à lui est dans le dossier `style/js/`, cependant je n'ai personnellement pas modifié de JavaScript, je me suis servis de celui existant pour quelques fonctions de base, comme l'affichage d'une boite de dialogue ou des messages d'informations pouvant être masqué à l'aide d'un clic.

bootstrap utilise le framework JavaScript Jquery.

3.1.3 Le HTML

Le HTML correspond aux vues, tous les fichiers contenant du HTML sont dans le dossier `views/`, les formulaires eux sont dans le dossier `views/forms`.

Les vues peuvent ne comporter que du HTML, comme c'est le cas dans les formulaires, mais ils peuvent également comporter des instructions simples de PHP comme des boucles ou des conditions.

Le dossier `views` contient également les fonctions d'affichage des documents ou une fonction permettant d'afficher une alerte.

```php
<?php
/**
 * Affiche une liste de documents
 * @param $xaDocuments Un tableau contenant tous les documents
 * @para $desc Vrai si la description des documents doit être affiché
 * @param $pseudo Vrai si les pseudos doivent être affichézs
 */
function listDocuments($xaDocuments, $desc=false, $pseudo=true) {
  echo '
  <table class="table table-striped"> ';
  foreach($xaDocuments as $doc) {
    echo '<tr>';
    echo '<td><a href="'.$doc['url'].'">'.$doc['titre'].'</a>   ';
    echo' </td>';
    if($desc) {
      echo '<td>'.$doc['description'].'</td>';
    }
    echo '<td>';
    $doc['motCle'] = array_unique($doc['motCle']); // on s'assure que les mots clés sont uniques
    foreach($doc['motCle'] as $keyword) {
      echo '<a href="search.php?k='.$keyword.'"><span class="label label-info">'.$keyword.'</span></a> ';
    }
    if($pseudo) {
      $doc['utilisateur'] = array_unique($doc['utilisateur']);
      echo '<td>';
      foreach($doc['utilisateur'] as $user) {
        echo '<a href="search.php?u='.$user.'"><span class="label">'.$user.'</span></a> ' ;
      }
      echo  '</td>';
    }
    echo '</td>';
    echo '</tr>';
  }
  echo '</table>';
}
```

Listing 3.2 – Fonction d'affichage d'une liste de documents

3.2 Connexions à la base de données : Les modèles

L'application comporte trois modèles : un pour les documents, un pour les termes, et un pour les utilisateurs. Ces trois fichiers contiennent des fonctions effectuant des requêtes et retournant un tableau avec les attributs, ceci afin de bien séparer le SQL du PHP.

Également présent dans le dossier `database`, le fichier `connect.php` contient les identifiants et les instructions de connexion à la base de données. Le fichier `database.php` quant à lui contient les requêtes de création et suppression de la base.

```
select document.numeroD, titre, url, motCle, description
from document

-- Jointures de l'intégralité du modèle
left join decrit on decrit.numeroD = document.numeroD
left join terme on terme.numeroT = decrit.numeroT
left join utilisateur on utilisateur.idUtilisateur = decrit.idUtilisateur
left join sauvegarde on sauvegarde.idUtilisateur = utilisateur.idUtilisateur
-- Selection de l'utilisateur
where utilisateur.idUtilisateur = &parametreIdUtilisateur
-- On joint également la table document
  and   document.numeroD = sauvegarde.numeroD
order by titre;
```

Listing 3.3 – Requête sélectionnant les documents d'un utilisateur donné

```
<?php
function insertDocument($xsTitle, $xsUrl, $xasKeywords, $xiIdUser, $xsDescription) {
  // si l'url n'existe pas dans la base, on l'insère
  if(!($urlFound = urlExist($xsUrl))) {
    $iIdDocument = uniqid(rand(), true);
    mysql_query("insert into document(numeroD, titre, url) values('$iIdDocument', '$xsTitle', '$xsUrl')") or die(mysql_error());
  } else { // sinon, on récupère son ID
    $iIdDocument = $urlFound->numeroD;
  }

  // maintenant on insère les mots clefs, s'ils n'existent pas déjà
  foreach($xasKeywords as $keyword) {
    if(!($keywordFound = keywordExist($keyword))) {
      $iIdTerme = uniqid(rand(), true);
      mysql_query("insert into terme(numeroT, motCle) values
        ('$iIdTerme', '".trim($keyword)."')") or die(mysql_error());
    } else {
      $iIdTerme = $keywordFound->numeroT;
    }

    if(!decritExist($iIdDocument, $iIdTerme)) {
      mysql_query("insert into decrit(numeroD, numeroT, idUtilisateur) values
        ('$iIdDocument', '$iIdTerme', '$xiIdUser')") or die(mysql_error());
    }

    if(!documentUserExist($iIdDocument, $xiIdUser)) {
      mysql_query("insert into sauvegarde(numeroD, idUtilisateur, description) values
        ('$iIdDocument', '$xiIdUser', '$xsDescription')") or die(mysql_error());
    }
  }
}
```

Listing 3.4 – Fonction d'insertion d'un document

3.3 Liaison des vues aux modèles : Les contrôleurs

Chaque contrôleur correspond aux pages que verra un utilisateur lambda dans l'URL, les contrôleurs sont en général assez simples, ils font appels au modèle puis font une inclusion de la vue, ceux sont eux qui feront les éventuels vérifications sur des paramètres, sur une connexion pour l'insertion de document etc...

```
<?php
$page = 'Saisir un document';
require_once('database/connect.php');
require_once('database/db_document.php');

include_once('views/header.php');
include_once('views/documentInsert.php');

if(isset($_POST['titre']) && isset($_POST['url']) && ↩
   isset($_POST['keywords'])) {
  $aKeywords = explode(';', $_POST['keywords']);
  foreach($aKeywords as $keyword) {
    $keyword = trim($keyword);
  }
  insertDocument($_POST['titre'], $_POST['url'], $aKeywords, $_SESSION['id'], ↩
     $_POST['description']);
}
```

Listing 3.5 – Contrôleur de l'insertion d'un document

Comme dit plus haut, c'est relativement simple, on inclus les vues et fonctions nécessaires, on vérifie qu'on a le droit d'être ici, on créer le tableau de mots clés, puis on insère le tout en faisant appel au modèle.

Chapitre 4

Résultats obtenus

Dans cette partie, nous allons regrouper quelques captures d'écrans montrant les résultats obtenus sur notre application Web. L'application est répartie en 5 fonctionnalités principales :
- Nuage de tags
- Recherche
- Liste des documents d'un utilisateur
- Affichage de « Mes documents »
- Création et suppression de la base de données

4.1 Nuage de tags et recherche

Figure 4.1 – Nuage de tags et recherche

Comme demandé dans les besoins, notre applications permet d'afficher un nuage de tags, celui-ci est visible sur la première partie de la figure 4.4. Les couleurs et les tailles des mots clés varient en fonction de leur nombre d'occurrence dans les différents documents.

Une recherche est également possible sur la page d'accueil, celle-ci se trouve en dessous du nuage de tag sur la figure 4.4. Il est possible de renseigner plusieurs mots clés différents, ceux-ci doivent être séparé par des espaces dans le champ de recherche.
Le résultat de la recherche s'affiche en dessous du champ de recherche.

R Ces deux fonctionnalités sont disponible dans être identifiés, tout le monde peut donc y avoir accès

4.2 Liste de tous les documents

Il est possible d'afficher la liste de tous les documents présents sur le site. Ceux-ci s'afficheront de la même manière que lors de l'affichage dans la recherche. Dans cette liste le clique sur un tag, renvoie sur un affichage de tous les documents associés à ce tag, le clique sur un utilisateur affiche tous les documents de cet utilisateur, enfin le clique sur le nom du document correspond à l'adresse web enregistrée en base de données.

Cette liste est disponible figure 4.2.

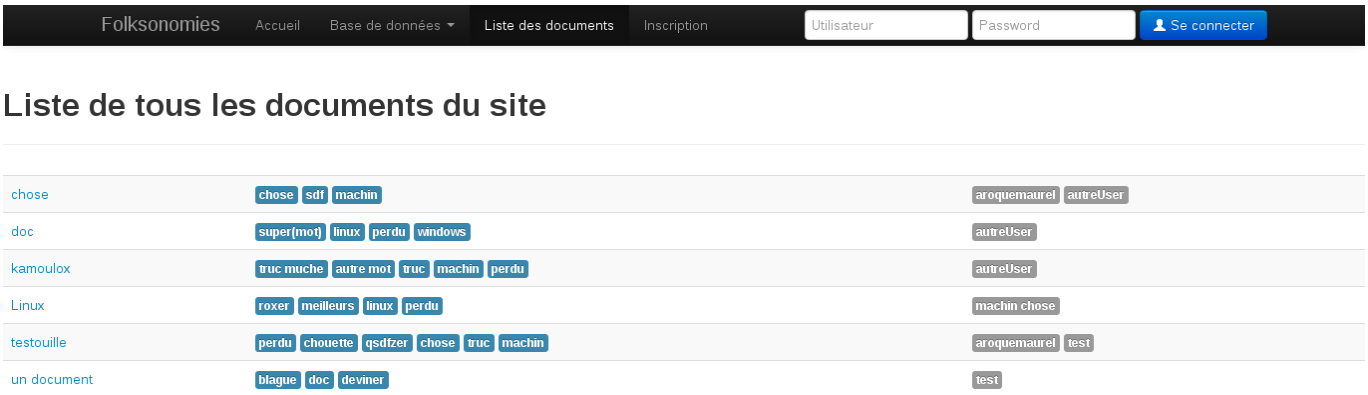

FIGURE 4.2 – Liste de tous les documents

R La liste de l'intégralité des documents du site est disponible à tout le monde, ainsi l'authentification n'est pas nécessaire.

4.3 Liste des documents d'un utilisateur

Comme le montre la figure 4.3 il est possible d'afficher les documents enregistrés par un utilisateur. Ces documents n'affichent pas la description mise par l'utilisateur, cette information étant considérée confidentielle.

FIGURE 4.3 – Liste des documents d'un utilisateur

4.4 Affichage de « Mes documents »

Lorsque l'on est connecté, on a la possibilité d'afficher ses propres documents. Ceux-ci s'affiche de la même manière que section 4.3 cependant, une information supplémentaire est affichée : la description mise par l'utilisateur.

Figure 4.4 – Liste de mes documents

4.5 Création et suppression de la base de données

Il est également possible de recréer ou de supprimer la base de données. Supprimer une base inexistante, ou créer une base déjà existante ne créera pas de bug, cependant l'action ne s'effectuera pas. [1].

Pour éviter les erreurs maladroites, une boite de dialogue s'affiche en cas de demande de suppression de la base de données, comme le montre la figure 4.5.

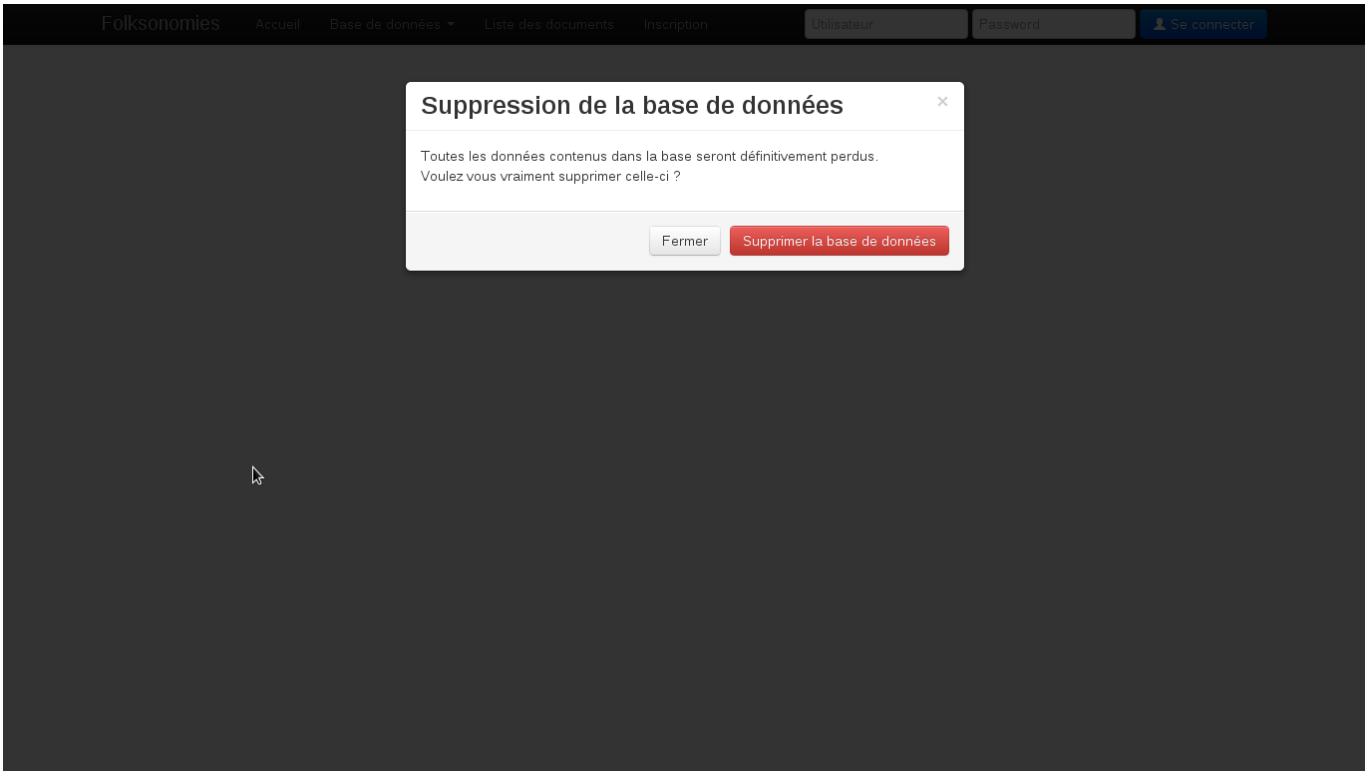

Figure 4.5 – Demande de suppression de la base de données

1. Utilisation de l'instruction mysql `if exist`

Annexe A

Table des figures

1.1 Modèle conceptuel de données de l'application 4

2.1 Affichage des demandes dans Redmine 6

4.1 Nuage de tags et recherche 12

4.2 Liste de tous les documents 13

4.3 Liste des documents d'un utilisateur 13

4.4 Liste de mes documents 14

4.5 Demande de suppression de la base de données 14

Annexe B

Liste des codes sources

3.1 Code CSS 8
3.2 Fonction d'affichage d'une liste de documents 9
3.3 Requête sélectionnant les documents d'un utilisateur donné 10
3.4 Fonction d'insertion d'un document 10
3.5 Contrôleur de l'insertion d'un document 11