

Antoine de ROQUEMAUREL ([antoine.de-roquemaurel@univ-tlse3.fr](mailto:antoine.de-roquemaurel@univ-tlse3.fr))  
Groupe 2.2

# Conception d'une application de folksonomies

---

Systèmes d'information et applications Web

---

# 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 tags. Il à été conçu 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 :

**scriptCreationBd.sql** Contient le script de création de la base de données, celui-ci contient un jeu d'essai. Il est également possible de supprimer le schéma, et de le créer via l'application.

**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.

---

# Table des matières

---

<b>1</b>	<b>Les besoins de l'application</b>	<b>4</b>
<b>2</b>	<b>Organisation du travail</b>	<b>5</b>
2.1	Un outil de gestion de projet : Redmine . . . . .	5
2.2	Un logiciel de versionnement : Git . . . . .	6
<b>3</b>	<b>Implémentation et conception</b>	<b>7</b>
3.1	Mise en forme : les vues . . . . .	7
3.1.1	Le CSS . . . . .	7
3.1.2	Le HTML . . . . .	8
3.2	Connexions à la base de données : Les modèles . . . . .	8
3.3	Liaison des vues aux modèles : Les contrôleurs . . . . .	9
<b>4</b>	<b>Résultats obtenus</b>	<b>11</b>
4.1	Affichage des séries et épisodes . . . . .	11
4.2	Recherche . . . . .	12
4.3	Ajout de séries et épisodes . . . . .	13
4.4	Valide HTML5 . . . . .	14
<b>A</b>	<b>Table des figures</b>	<b>15</b>
<b>B</b>	<b>Liste des codes sources</b>	<b>15</b>

---

# 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 ??? est présent le nouveau modèle.

**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 en fonction d'un ou plusieurs tags** On peut rechercher un document en fonction d'un ou plusieurs 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 personnaliser 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.

# 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

The screenshot displays the Redmine web application interface. At the top, there's a navigation bar with links like 'Accueil', 'Ma page', 'Projets', 'Administration', and 'Aide'. Below this, a search bar and a dropdown menu for 'superVOD' are visible. The main content area is titled 'Demandes' and shows a list of tasks. The tasks are organized into a table with columns: '#', 'Tâche parente', 'Tracker', '% réalisé', 'Statut', 'Sujet', 'Assigné à', and 'Mis-à-jour'. The tasks are listed with their IDs, parent tasks, trackers, progress bars, statuses, subjects, assignees, and last update times. A sidebar on the right contains links to 'Demandes', 'Rapports personnalisés', and 'Rapport'.

#	Tâche parente	Tracker	% réalisé	Statut	Sujet	Assigné à	Mis-à-jour
307		User Story		Terminée	Interface		12-04-2013 18:49
308		User Story		Terminée	Recherche		13-04-2013 22:00
309	User-Story-#308: Recherche	User Story		Terminée	» Série		13-04-2013 21:59
334	User-Story-#309: Série	Tache		Terminée	» Ajouter public visé		13-04-2013 21:59
335	User-Story-#309: Série	Tache		Terminée	» Formulaire de recherche		13-04-2013 15:46
336	User-Story-#309: Série	Tache		Terminée	» Traitement formulaire		13-04-2013 15:46
310	User-Story-#308: Recherche	User Story		Terminée	» Épisode		13-04-2013 22:00
312	User-Story-#308: Recherche	User Story		Terminée	Accès privé		21-04-2013 15:10
322	User-Story-#312: Accès privé	User Story		Terminée	» Insertion épisode		20-04-2013 11:44
330	User-Story-#322: Insertion épisode	User Story		Terminée	» Upload image		20-04-2013 11:44
337	User-Story-#322: Insertion épisode	Tache		Terminée	» Formulaire		13-04-2013 22:01
338	User-Story-#322: Insertion épisode	Tache		Terminée	» Insertion des données		13-04-2013 22:01
339	User-Story-#322: Insertion épisode	Tache		Terminée	» Vérifications des données		13-04-2013 22:01
323	User-Story-#312: Accès privé	User Story		Terminée	» Insertion série		20-04-2013 11:44
329	User-Story-#312: Accès privé	Tache		Terminée	» Vérifications		12-04-2013 23:44
313		Document		En cours	Rapport		21-04-2013 13:45
324		User Story		Terminée	Liste série		13-04-2013 22:21
325	User-Story-#324: Liste série	Tache		Terminée	» Menu séries		13-04-2013 22:02
326	User-Story-#324: Liste série	Tache		Terminée	» Affichage épisodes		12-04-2013 23:45
327	User-Story-#324: Liste série	Tache		Terminée	» Durée pour les humains		13-04-2013 22:11
328	User-Story-#324: Liste série	Tache		Terminée	» Editer Hello World		13-04-2013 22:21

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/terminées/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 intérêts, tout d'abord, je pouvais travailler de plusieurs ordinateurs en parallèle sur le projet sans me soucier de fusionner le travail<sup>1</sup>.

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

Enfin, toutes les modifications 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 a régressé.

---

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

---

# Implémentation et conception

---

La conception du projet est l'étape la plus importante, ainsi j'ai préféré repenser à notre manière le projet, je suis parti sur une architecture respectant le patron de conception MVC<sup>1</sup>, 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<sup>2</sup> 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`.

```
1 .menuSeries {  
    padding-left: 30px; /* On ne colle pas le menu à gauche de l'écran */  
3 }  
#series {  
5     float: left; /* flottement lors de la recherche d'une série */  
6 }  
7  
/* Taille/affichage du carousel */  
9 .carousel h1 {  
    color: #d5d5d5;  
11 }  
12 .carousel {  
13     height: 250px;  
    width: 800px;  
15     background-color: black;  
    margin: auto;  
17     margin-bottom: 15px;  
18 }
```

Listing 3.1 – Exemple d'une partie du 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 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.

```

1 <?php
2 /**
3  * Affiche une liste de documents
4  * @param $xaDocuments Un tableau contenant tous les documents
5  * @para $desc Vrai si la description des documents doit être affiché
6  * @param $pseudo Vrai si les pseudos doivent être affichés
7  */
8 function listDocuments($xaDocuments, $desc=false, $pseudo=true) {
9     echo '
10     <table class="table table-striped"> ';
11     foreach($xaDocuments as $doc) {
12         echo '<tr>';
13         echo '<td><a href="'. $doc['url'] . '>'. $doc['titre'] . '</a>&nbsp;&nbsp; ';
14         echo '</td>';
15         if($desc) {
16             echo '<td>'. $doc['description'] . '</td>';
17         }
18         echo '<td>';
19         $doc['motCle'] = array_unique($doc['motCle']); // on s'assure que les mots
20             clés sont uniques
21         foreach($doc['motCle'] as $keyword) {
22             echo '<a href="search.php?k="'. $keyword . '><span class="label
23                 label-info">'. $keyword . '</span></a> ';
24         }
25         if($pseudo) {
26             $doc['utilisateur'] = array_unique($doc['utilisateur']);
27             echo '<td>';
28             foreach($doc['utilisateur'] as $user) {
29                 echo '<a href="search.php?u="'. $user . '><span
30                     class="label">'. $user . '</span></a> ';
31             }
32             echo '</td>';
33         }
34         echo '</td>';
35         echo '</tr>';
36     }
37     echo '</table>';
38 }

```

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.

```

1 select document.numeroD, titre, url, motCle, description
  from document
3
4 -- Jointures de l'intégralité du modèle
5 left join decrit on decrit.numeroD = document.numeroD
6 left join terme on terme.numeroT = decrit.numeroT
7 left join utilisateur on utilisateur.idUtilisateur = decrit.idUtilisateur
8 left join sauvegarde on sauvegarde.idUtilisateur = utilisateur.idUtilisateur
9 -- Selection de l'utilisateur
10 where utilisateur.idUtilisateur = &parametreIdUtilisateur
11 -- On joint également la table document
12 and document.numeroD = sauvegarde.numeroD
13 order by titre;
```

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

```

1 <?php
function insertDocument($xsTitle, $xsUrl, $xasKeywords, $xiIdUser, <←
    $xsDescription) {
3     // si l'url n'existe pas dans la base, on l'insère
4     if(!($urlFound = urlExist($xsUrl))) {
5         $iIdDocument = uniqid(rand(), true);
6         mysql_query("insert into document(numeroD, titre, url) <←
7             values('$iIdDocument', '$xsTitle', '$xsUrl')") or die(mysql_error());
8     } else { // sinon, on récupère son ID
9         $iIdDocument = $urlFound->numeroD;
10    }
11
12    // maintenant on insère les mots clefs, s'ils n'existent pas déjà
13    foreach($xasKeywords as $keyword) {
14        if(!($keywordFound = keywordExist($keyword))) {
15            $iIdTerme = uniqid(rand(), true);
16            mysql_query("insert into terme(numeroT, motCle) values
17                ('$iIdTerme', '".trim($keyword)."'") or die(mysql_error());
18        } else {
19            $iIdTerme = $keywordFound->numeroT;
20        }
21
22        if(!decritExist($iIdDocument, $iIdTerme)) {
23            mysql_query("insert into decrit(numeroD, numeroT, idUtilisateur) values
24                ('$iIdDocument', '$iIdTerme', '$xiIdUser')") or die(mysql_error());
25        }
26
27        if(!documentUserExist($iIdDocument, $xiIdUser)) {
28            mysql_query("insert into sauvegarde(numeroD, idUtilisateur, description) <←
29                values
30                ('$iIdDocument', '$xiIdUser', '$xsDescription')") or die(mysql_error());
31        }
32    }
33 }
```

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. . .

```
1 <?php
2 $page = 'Saisir un document';
3 require_once('database/connect.php');
4 require_once('database/db_document.php');
5
6 include_once('views/header.php');
7 include_once('views/documentInsert.php');
8
9 if(isset($_POST['titre']) && isset($_POST['url']) && ↵
10     isset($_POST['keywords'])) {
11     $aKeywords = explode(';', $_POST['keywords']);
12     foreach($aKeywords as $keyword) {
13         $keyword = trim($keyword);
14     }
15     insertDocument($_POST['titre'], $_POST['url'], $aKeywords, $_SESSION['id'], ↵
16         $_POST['description']);
17 }
```

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 crée le tableau de mots clés, puis on insère le tout en faisant appel au modèle.

## 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 3 fonctionnalités :

- L'affichage des séries et épisodes présents
- La recherche de série ou épisodes
- L'ajout de série ou épisodes

L'ajout d'épisodes implique la possibilité de se connecter afin que seul l'administrateur puisse utiliser l'administration.

**R** Le couple login/mot de passe de l'application est admin/admin

### 4.1 Affichage des séries et épisodes

	Titre	Numéro	Année	Réalisateur	Durée	Limite age
Saison 1						
	Noel mortel	1	1990	David Silverman	25mn	0
	Bart le genie	2	1990	David Silverman	25mn	0

FIGURE 4.1 – Page d'accueil de l'application

La figure 4.1 nous montre la page d'accueil du site, avec la possibilité de sélectionner une série via le menu de gauche, affichant ainsi tous ses épisodes. On peut remarquer sur cette figure que le site possède un thème qui est commun à tout le site : la barre de navigation en haut de l'écran permet d'accéder d'un clique à chacune des fonctionnalités présentes sur le site.

4.2 Recherche

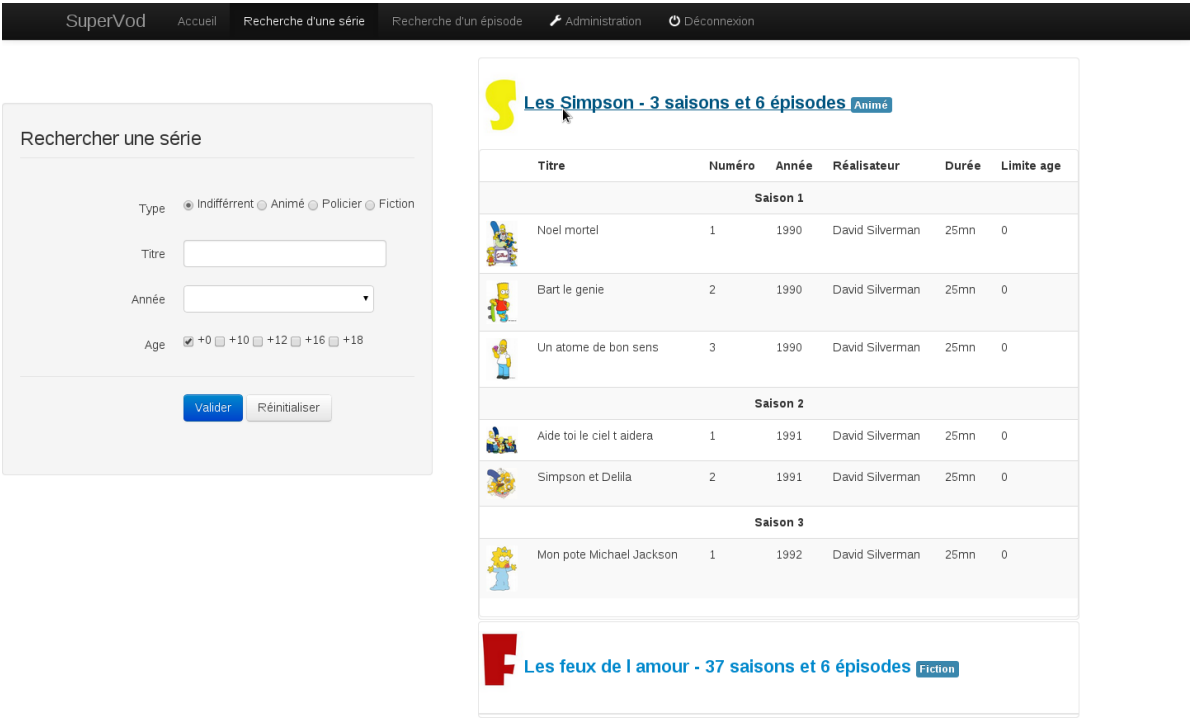


FIGURE 4.2 – Recherche d’une série

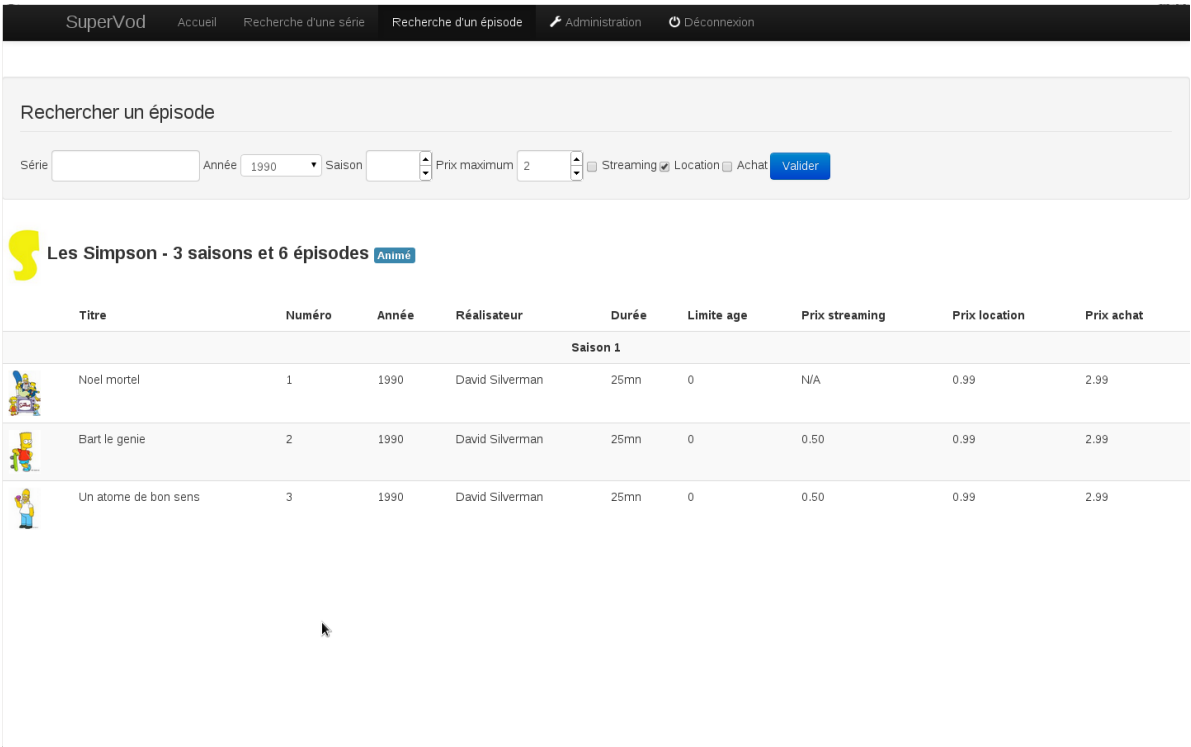


FIGURE 4.3 – Recherche d’un épisode

Les figures 4.2 et 4.3 nous montrent respectivement la recherche d'une série et la recherche d'un épisode.

Ces deux fonctionnalités sont assez similaires, bien que les critères de recherches soit différents. Ainsi, la recherche d'une série affiche les séries répondants au critères renseignés <sup>1</sup>, ainsi que la possibilité d'affiche les épisodes correspondants de la série.

La recherche d'un épisode quand à elle à une mise en forme différente, étant donné qu'il y a plus d'informations à afficher pour un épisode, cependant le principe reste le même via les critères de recherche <sup>2</sup>

## 4.3 Ajout de séries et épisodes

The screenshot shows the 'Ajouter un épisode' (Add episode) form in the SuperVod application. The form is located under the 'Administration' menu. It contains the following fields and options:

- Série:** A dropdown menu with 'Doctor Who' selected.
- Titre:** A text input field containing 'An Unearthly Child'.
- Numéro:** A numeric input field with '1'.
- Année:** A year selection dropdown with '1963'.
- Saison:** A season selection dropdown with '1'.
- Possibilité d'achat:** A section with three checkboxes and corresponding price inputs:
  - Location:** Checked, with a price of 1,5.
  - Achat:** Not checked, with an empty price field.
  - Streaming:** Checked, with a price of 0,99.
- Réalisateur:** A text input field containing 'Waris Hussein'.
- Durée:** A numeric input field.
- Image:** A file upload button labeled 'Choisissez un fichier' and a status message 'Aucun fichier choisi'.
- Limite d'age:** A dropdown menu with '0' selected.

FIGURE 4.4 – Administration

La figure 4.4 nous montre la partie administration, ici l'ajout d'un nouvel épisode à la série *Doctor Who*, tous les champs présents précédemment peuvent être renseignés, cependant seuls les champs Série, Titre, Numéro et Saison sont obligatoires, les autres sont facultatifs.

Il est également possible de sélectionner une image pour la série depuis notre ordinateur via un champ d'upload.

Nous n'avons pas fait de capture d'écrans de l'ajout d'une série, cependant le principe est le même bien que les champs soient moins nombreux, pour une série, les champs Type et Nom sont obligatoires.

---

1. Une série peut être cherché par son type, son titre, ses années de diffusion ainsi que son age  
 2. Un épisode peut être trouvé grâce à sa série, son année de diffusion, sa saison, le prix maximum auquel on veut l'acheter, ainsi que sa disponibilité (Streaming, Location, Achat)

## 4.4 Valide HTML5

**W3C® Markup Validation Service**  
Check the markup (HTML, XHTML, ...) of Web documents

Jump To: [Notes and Potential Issues](#) [Congratulations · Icons](#)

**This document was successfully checked as HTML5!**

<b>Result:</b>	Passed, <b>2 warning(s)</b>		
<b>Address :</b>	<input type="text" value="http://dev.joohoo.fr/dev/supervod/index.php"/>		
<b>Encoding :</b>	utf-8	<input type="button" value="(detect automatically)"/>	
<b>Doctype :</b>	HTML5	<input type="button" value="(detect automatically)"/>	
<b>Root Element:</b>	html		

The W3C validators rely on community support for hosting and development. [Donate](#) and help us build better tools for a better web. 4062

**Options**

☐ Show Source     
 ☐ Show Outline     
 ☒ List Messages Sequentially   
 ☒ Group Error Messages by Type  
☐ Validate error pages     
 ☐ Verbose Output     
 ☐ Clean up Markup with HTML-Tidy

[Help](#) on the options is available.

**Notes and Potential Issues**

The following notes and warnings highlight missing or conflicting information which caused the validator to perform some guesswork prior to validation, or other things affecting the output below. If the guess or fallback is incorrect, it could make validation results entirely incoherent. It is *highly recommended* to check these potential issues, and, if necessary, fix them and re-validate the document.

**Using experimental feature: HTML5 Conformance Checker.**

The validator checked your document with an experimental feature: *HTML5 Conformance Checker*. This feature has been made available for your convenience, but be aware that it may be unreliable, or not perfectly up to date with the latest development of some cutting-edge technologies. If you find any issues with this feature, please [report them](#). Thank you.

FIGURE 4.5 – HTML5 validator – Passed

La figure 4.5 montre simplement notre site web passant la validation du w3c<sup>3</sup> avec la norme HTML5.

---

## Table des figures

---

2.1	Affichage des demandes dans Redmine . . . . .	5
4.1	Page d'accueil de l'application . . . . .	11
4.2	Recherche d'une série . . . . .	12
4.3	Recherche d'un épisode . . . . .	12
4.4	Administration . . . . .	13
4.5	HTML5 validator – Passed . . . . .	14

---

## Liste des codes sources

---

3.1	Exemple d'une partie du CSS . . . . .	7
3.2	Fonction d'affichage d'une liste de documents . . . . .	8
3.3	Requête sélectionnant les documents d'un utilisateur donné . . . . .	9
3.4	Fonction d'insertion d'un document . . . . .	9
3.5	Contrôleur de la liste des documents . . . . .	10