

Antoine de ROQUEMAUREL (antoine.de-roquemaurel@univ-tlse3.fr)

Fabrice VALLEIX (valleix.fabrice@gmail.com)

Groupe 2.2

Conception d'une application Web – SuperVod

Systèmes d'information et applications Web

Avant-propos

Ce dossier comporte les différentes de réalisation de l'application Web SuperVod, site permettant de répertorier des séries.

Il à été conçu par Antoine de ROQUEMAUREL et Fabrice VALLEIX 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 était organisée comme ceci :

scriptCreationBd.sql Contient le script de création de la base de données, le schéma n'a pas été changé, cependant MySQL étant sensible à la casse, des erreurs avait lieu avec le script de création qui nous étai fournis, ainsi par convention, toutes nos tables sont en minuscule.

superVod/ 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/superVod/>. 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 `superVod/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.

Le site disposant d'un système d'upload d'image, afin que celui-ci fonctionne correctement, il est nécessaire que le dossier **images** possèdent les droits d'écritures pour le site web, soit les droits 777 dans le système Unix, ou appartenir au groupe **www-data**.

Pour se connecter à la partie administration, les identifiants sont :

Login admin

Mot de passe admin

Table des matières

1	Les besoins de l'application	4
1.1	Affichage des séries et épisodes	4
1.2	Recherche de séries et épisodes	4
1.3	Ajout de série et épisode	4
2	Organisation du travail d'équipe	5
2.1	Un outil de gestion de projet : Redmine	5
2.2	Un logiciel de versionnement : Git	6
2.3	Répartition des tâches	6
3	Implémentation et conception	7
3.1	Mise en forme : les vues	7
3.1.1	Le CSS	7
3.1.2	Le JavaScript	8
3.1.3	Le HTML	8
3.2	Connexions à la base de données : Les modèles	9
3.3	Liaison des vues aux modèles : Les contrôleurs	10
4	Résultats obtenus	11
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
A	Table des figures	15
B	Liste des codes sources	15

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 répertorier des séries et leurs épisodes associés.

Pour cela le schéma de la base de données et un jeu d'essai nous à été fournis afin de pouvoir commencer rapidement le développement du site.

1.1 Affichage des séries et épisodes

La possibilité d'afficher dynamiquement les séries et leurs épisodes avec ainsi pour chaque série son titre, son nombre de saison et son nombre d'épisodes ainsi que son type, pour chaque épisode d'une série son titre, sa saison, son numéro d'épisode son année de production, son réalisateur sa durée et sa limite d'âge.

Étant donné que le site est voué à disposé d énormément de séries différentes, plutôt que de tout afficher sur une page, nous avons souhaités ajouter un menu permettant de sélectionner la série. Cf image ?? page ??.

1.2 Recherche de séries et épisodes

Le site doit permet d'effectuer des recherches, d'une part des recherches d'épisodes, en fonction d'une partie du titre, ses année de production, son type, son age minimum, toutes les séries trouvés grâce à la recherche permette d'afficher les épisodes de la série.

Mais il doit également être possible de chercher un épisode en fonction de la série concernée, de son année de diffusion, de sa saison et de ses différents prix d'achats maximum (Streaming, location ou achat) dans le cas où on souhaite acheter un épisode.

Des captures d'écrans de recherches sont disponibles image ?? et ?? page ??.

1.3 Ajout de série et épisode

Afin que le dynamisme du site ai un intérêt, il devait être possible d'ajouter des séries ou des épisodes dynamiquement. Ainsi une partie administration à été créé afin que seul l'administrateur puisse ajouter des séries. Lors de l'ajout d'une série ou d'un épisode, une image peut être téléchargée sur le site à l'aide d'un formulaire d'envoi.

Organisation du travail d'équipe

Pour ce projet, nous étions deux à travailler dessus, ainsi nous avons utilisé plusieurs techniques afin de se coordonner et de limiter les problèmes. Ceci n'est pas notre premier projet ensemble, notre travail en fut simplifié.

2.1 Un outil de gestion de projet : Redmine

The screenshot displays the Redmine web application interface for the 'superVOD' project. The main section shows a list of 'Demandes' (Requests) with the following columns: #, Tâche parente, Tracker, % réalisé, Statut, Sujet, Assigné à, and Mis-à-jour. The table contains 21 entries, with the last entry (ID 327) highlighted in red, indicating it is the current task. The sidebar on the right provides additional navigation options like '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	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	Document		En cours	Rapport		21-04-2013 13:45
324	User Story	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, nous avons utilisé *Redmine*, une plateforme web de gestion de projet (Cf figure 2.1). Elle nous a permis de simplifier le travail, et de ne rien oublier.

En effet, nous pouvons créer des tâches, signaler qu'elles sont en cours/terminées/en tests, leur donner des dates limites, les affecter à une personne etc... Ainsi lorsque l'un de nous commençait une tâche, il le signalait sur le *redmine*, ce qui permettait de tenir au courant son binôme de ses actions et de l'avancée du projet.

2.2 Un logiciel de versionnement : Git

Afin de limiter les problèmes du travail collaboratif, nous avons utilisé un logiciel de versionnement Git. Il a deux intérêt, tout d'abord, nous pouvons travailler à deux en parallèle sur le projet sans se soucier de fusionner notre travail¹.

D'autre part, tous les logs étant enregistrés, nous pouvons savoir qui à fait quoi et quel jour, cela permet de voir également 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.

2.3 Répartition des tâches

Nous nous sommes répartis les tâches afin de travailler efficacement, en fonction de nos domaines d'affinités et d'expertise, ainsi Fabrice s'occupait plutôt de la partie base de données et SQL pendant qu'Antoine architecturait l'application afin qu'elle soit la plus cohérente possible, le reste du projet était fait sans répartition particulière, en fonction des besoins et de l'envie de chacun.

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 nous avons préférés repenser à notre manière le projet, nous sommes partis sur une architecture respectant plus le patron de conception MVC¹, 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, nous avons choisis d'utiliser un framework² 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 nous avons d'une part le CSS fourni par *bootstrap*, auquel nous n'avons pas touché, d'autre part, notre 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 JavaScript

Les bibliothèques JavaScript utilisées par *bootstrap* sont présentes dans `style/js`, celle-ci nous n'y avons pas touchés, le code JavaScript que nous avons développé étaient toujours très simple, ceci dû au travail de *bootstrap*, ainsi il est présent en fin de page HTML le cas échéant.

```

// quand on change l'état de la checkbox, si elle devient cochée
2 // On met le input en read/write, sinon on grise
// Pour la location mais le principe est le même pour les autres
4 $("#typeLocation").on("change", function() {
    if($("#typeLocation").is(':checked')) {
6         $('#prixLocation').removeAttr('readonly')
    } else {
8         $('#prixLocation').attr('readonly', 'readonly')
    }
10 });

```

Listing 3.2 – Code javascript grisant le champ de prix lors de l'ajout d'un épisode

3.1.3 Le HTML

Le HTML correspond aux vues, tous nos 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 de série et épisode servant uniquement à l'affichage de ceux-ci.

```

<div id="formSearch">
2   <?php include('views/forms/rechercheSerie.php'); // formulaire de recherche?>
</div>

4
<?php
6 echo '<div id="series">';
echo '<div id="monaccordeon">';
8 $i = 0;
foreach($series as $serie) { ?>
10     <!-- On utilise un accordien de Bootstrap pour éviter de surcharger la
        page lorsqu'on affiche pour chaque série ses épisodes associés -->
    <div class="accordion-group">
12     <div class="accordion-heading" data-toggle="collapse"
        data-parent="#monaccordeon" data-target="#item'.$i.'">
        <a style="cursor: pointer">
14         <?php afficherSerie($serie); // on affiche la série courante ?>
        </a>
    </div>
16     <div id="item'.$i.'" class="collapse accordion-group">
18         <?php afficherListeEpisodes($episodes[$i]); // la liste des épisodes de
            la série courante
        ?>
20     </div>
    </div>
22 <?php
    ++$i;
24 }
?>
26 </div>

```


</div>

Listing 3.3 – Vue de la page permettant de rechercher une série

3.2 Connexions à la base de données : Les modèles

L'application ne comporte que deux modèles : un pour les séries et un pour les épisodes, ces deux 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.

```

1  -- Retourne toutes les séries avec pour chaque série le nombre de saison
  -- et le nombre d'épisodes
3  select distinct noms, series.image, series.cs, types, max(saison) AS ←
    nb_saisons, count(numero) as nb_episodes
  from series
5  left join episodes
  on series.cs = episodes.cs
7  group by types, noms

```

Listing 3.4 – Requête sélectionnant toutes les séries

```

1  <?php
  function searchEpisodes($psSerie, $piAnnee, $piSaison, $piPrix, $paType) {
3    $return = array();
    $whereType = '';
5    $bAnnee = isset($piAnnee) && $piAnnee != '';
    /* construction de la requête */
7    $requete = 'select distinct series.cs, episodes.ce as ep, titre, saison, ←
      numero, annee, realisateur, '
    $requete .= 'de,lim,episodes.image as epImage, series.image as seImage, '
9    // selection du prix des fichiers
    $requete .= '(select prix from fichiers,episodes where fichiers.ce = ep and ←
      type=\'S\' and episodes.ce=fichiers.ce) as prixStream, '
11   $requete .= '(select prix from fichiers,episodes where fichiers.ce = ep and ←
      type=\'L\' and episodes.ce=fichiers.ce) as prixLocation, '
    $requete .= '(select prix from fichiers,episodes where fichiers.ce = ep and ←
      type=\'A\' and episodes.ce=fichiers.ce) as prixAchat '
13   // jointures
    $requete .= 'from episodes left join fichiers on fichiers.ce = episodes.ce
15   join series on series.cs = episodes.cs';
    // Recherche du nom
    $requete .= 'where noms LIKE LOWER(\'%.strtolower($psSerie).\'%\')';
    /* en fonction de ce qu'a rentré l'utilisateur, on ajoute les différents ←
      critères */
19   if($bAnnee) {
      $requete .= ' AND annee = \''.$piAnnee.'\'';
21   }
    $nbType = 0;
23   foreach($paType as $type) {
      if($type) {
25         $whereType .= ' AND fichiers.type <> \''.$type.'\''; // Les types ←
          d'achat possibles
          ++$nbType;
27       }
    }
29   if($nbType != 3) //si rien n'est coché, on ne filtre pas

```

```

31     $requete .= $whereType;
33
34     if($piSaison != '')
35         $requete .= ' AND saison=\''.$piSaison.'\'';
36
37     if($piPrix != '')
38         $requete .= ' AND prix<\'$piPrix\'';
39     $result = mysql_query($requete) or die(mysql_error()); // on execute la requete
40     while($object = mysql_fetch_object($result)) {
41         $return[$object->cs][] = $object; // on met tout dans un tableau
42     }
43
44     return $return;
45 }

```

Listing 3.5 – Requête recherchant des épisodes

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 la partie privée etc...

```

<?php
2 $page = 'Accueil'; // Titre de la page
include_once('views/header.php');
4 require_once('views/episodes.php');
require_once('views/series.php');
6 require_once('functions/util.php');

8 require_once('database/db_series.php');
require_once('database/db_episodes.php');
10 $series = getAllSeriesComplete();
$iSerieActuel = 0;

12
13 if(!isset($_GET['serie'])) {
14     $iSerieActuel = 1;
15 } else {
16     $iSerieActuel = $_GET['serie'];
17 }

18
19 if(isset($_GET['p'])) {
20     switch($_GET['p']) {
21         case "connect":
22             include_once('connexion.php');
23             break;
24         case "disconnect":
25             include_once('deconnexion.php');
26             break;
27     }
28 }

30 include_once('views/index.php');
include_once('views/footer.php');
32 ?>

```

Listing 3.6 – Contrôleur de la page d'accueil

Comme dit plus haut, c'est relativement simple, on inclus les vues et fonctions nécessaires, on demande l'obtention des séries, on fait quelques vérifications puis on affiche la vue de la page d'accueil.

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

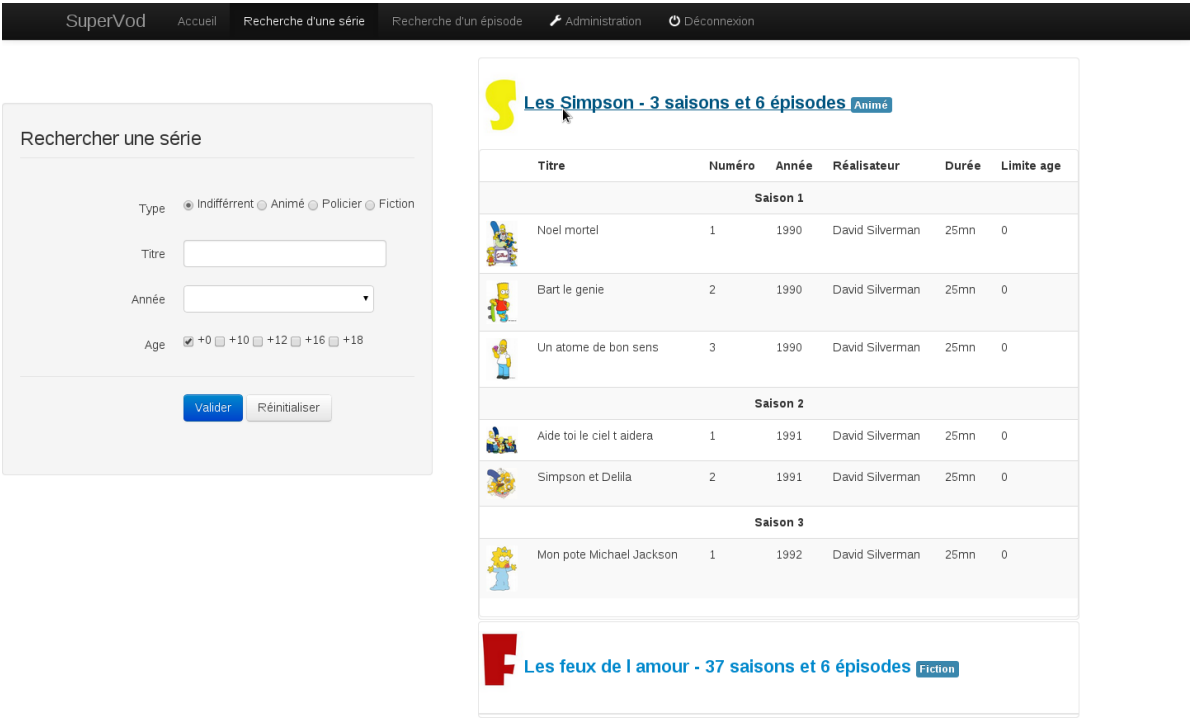


FIGURE 4.2 – Recherche d’une série

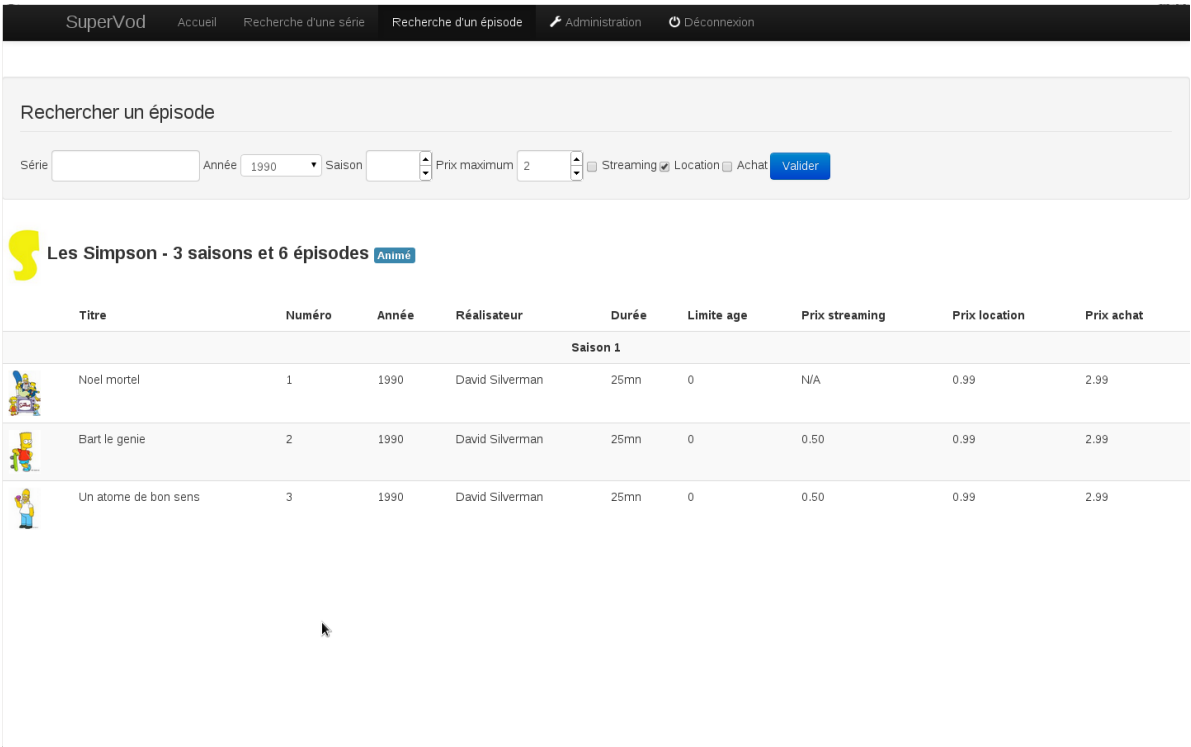


FIGURE 4.3 – Recherche d’un épisode

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 ¹, 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 ²

4.3 Ajout de séries et épisodes

The screenshot shows the 'Administration' section of the SuperVod application. At the top, there is a navigation bar with links: 'SuperVod', 'Accueil', 'Recherche d'une série', 'Recherche d'un épisode', 'Administration' (active), and 'Déconnexion'. Below the navigation bar, there are two buttons: 'Ajouter une série' and 'Ajouter un épisode'. The 'Ajouter un épisode' form is displayed, titled 'Ajouter un épisode'. The form contains the following fields:

- Série:** A dropdown menu with 'Doctor Who' selected.
- Titre:** A text input field containing 'An Unearthly Child'.
- Numéro:** A text input field containing '1'.
- Année:** A dropdown menu with '1963' selected.
- Saison:** A dropdown menu with '1' selected.
- Possibilité d'achat:** A section with three options:
 - Location:** A checkbox that is checked, followed by a price input field containing '1,5'.
 - Achat:** A checkbox that is unchecked, followed by a price input field.
 - Streaming:** A checkbox that is checked, followed by a price input field containing '0,99'.
- Réalisateur:** A text input field containing 'Waris Hussein'.
- Durée:** A text input field.
- Image:** A button labeled 'Choisissez un fichier' and a text label 'Aucun fichier choisi'.
- Limite d'âge:** 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!

Result:	Passed, 2 warning(s)		
Address :	<input type="text" value="http://dev.joohoo.fr/dev/supervod/index.php"/>		
Encoding :	utf-8	<input type="button" value="(detect automatically)"/>	
Doctype :	HTML5	<input type="button" value="(detect automatically)"/>	
Root Element:	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 w3c³ 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	Code javascript grisant le champ de prix lors de l'ajout d'un épisode	8
3.3	Vue de la page permettant de rechercher une série	8
3.4	Requête sélectionnant toutes les séries	9
3.5	Requête recherchant des épisodes	9
3.6	Contrôleur de la page d'accueil	10