De Sousa Soares Kevin I.Fa-P3A

TPI 2017  Documentation Technique

18 juin 2017

MyMoviesList

Table des matières

[2 Introduction 2](#_Toc485567464)

[2.1 Motivations 2](#_Toc485567465)

[2.2 Objectifs 2](#_Toc485567466)

[2.3 Description détaillée 2](#_Toc485567467)

[3 Analyse de l’existant 3](#_Toc485567468)

[4 Analyse fonctionnelle 4](#_Toc485567469)

[4.1 Plan du site 4](#_Toc485567470)

[4.2 Maquettes 4](#_Toc485567471)

[4.3 Fonctionnalités 9](#_Toc485567472)

[4.4 Logiciels utilisés 9](#_Toc485567473)

[5 Analyse Organique 10](#_Toc485567474)

[5.1 Modèle conceptuel 10](#_Toc485567475)

[5.2 Fonctionnement détaillé 11](#_Toc485567476)

[5.3 Arborescence 12](#_Toc485567477)

[5.4 OMDb API 13](#_Toc485567478)

[5.5 Sécurité des pages 13](#_Toc485567479)

[5.6 Base de données 14](#_Toc485567480)

[5.6.1 Modèle physique de la base de données 14](#_Toc485567481)

[5.6.2 Description des tables 15](#_Toc485567482)

[5.6.3 Requêtes SQL 16](#_Toc485567483)

[5.7 Gestion dans le contrôleur 18](#_Toc485567484)

[6 Tests 20](#_Toc485567485)

[7 Conclusion 22](#_Toc485567486)

[7.1 Comparaison planning 22](#_Toc485567487)

[7.2 Améliorations futures 23](#_Toc485567488)

[7.3 Choix et changements 23](#_Toc485567489)

[7.4 Bilan final 23](#_Toc485567490)

[8 Glossaire 24](#_Toc485567491)

[9 Bibliographie 25](#_Toc485567492)

[9.1 Source 25](#_Toc485567493)

[9.2 Tableau de figure 26](#_Toc485567494)

[10 Annexes 26](#_Toc485567495)

# Introduction

Je suis en dernière année de mon cursus je dois donc effectuer un travail dans le cadre de mon TPI afin d’obtenir mon CFC.

## Motivations

J’ai choisi de faire un site web sur le thème du cinéma car je me sens plus à l’aise en web et que cela se prêtait plus pour mon projet parce que j’ai décidé d’utiliser une API. Pour mon sujet les films ont été un choix évident car on peut faire beaucoup de site sur ce sujet et parce que j’aime regarder des films. J’aime aussi garder une trace de ce que j’ai regardé et ce que je dois regarder, l’idée de faire un site avec une liste de films vus et une liste de films à voir m’est venue donc assez facilement. L’API me permet en plus d’avoir une grande base de données avec beaucoup de film.

## Objectifs

L’objectif est de créer un site web permettant aux utilisateurs de créer une librairie de films qu’ils souhaitent voir ou qu’ils ont déjà vu. Les utilisateurs pourront ajouter les films dans leur liste de films en écrivant un commentaire sur le film. Les utilisateurs peuvent voir les listes de film et les commentaires des autres utilisateurs.

## Description détaillée

Seul un utilisateur connecté peut avoir une liste. Les listes sont prédéfinies dans l’application (à voir, déjà vu, etc.). Le site doit donc permettre à un utilisateur de se créer un compte, de se connecter et de se déconnecter. Une fois le compte créé, l’utilisateur pourra se connecter et ajouter des films à ses différentes listes (déjà vu, veut voir). Il pourra également consulter/éditer/supprimer les éléments de ses listes pour faire passer les films d’une liste à l’autre ou tout simplement l’enlever de la liste.

Un utilisateur non connecté pourra regarder les informations sur les films. Pour voir les infos d’un film on pourra rechercher le film grâce à une fonctionnalité de recherche. La recherche de film se fera sur la base de l’API OMDB qui permet de récupérer des informations de films et de les relier ensuite à IMDB. De ce fait, les films ne seront pas stockés dans notre base de données. Seul l’id IMDB du film sera stocké afin de garder le lien vers le film existant.

La liste de film dispose de pagination et l’utilisateur aura la possibilité de trier la liste par films les plus vus, par nombre de présence dans une liste, etc. En cliquant sur un film dans la liste de films on est dirigé vers une page avec les informations du film en question.

On pourra voir depuis cette page également les commentaires ainsi que le ranking global du film

# Analyse de l’existant

Il existe pas mal de sites de gestions de films et même des applications pour smartphone. L’un des sites les plus connu est *senscritique.com*. Le site permet à chacun de noter un film et de rédiger une critique sur ce film que les autres utilisateurs pourront lire. Le site permet aussi de créer un nombre infini de listes de films et de les rendre publiques ou privés, les listes dans ce site sont faites par l’utilisateur qui choisit comment l’appeler et ce qu’il met dedans. On peut donc voir sur le site des listes par genre de film ou avec des thématiques plus spécifiques.



1 Screenshot du site SensCritique où l’on peut voir une liste créé par un utilisateur

SensCritique est basé sur le partage de film, de livre, de musique et de jeux et sur la critique de ces œuvres. Ce que je souhaite faire c’est un gestionnaire de film qui ressemble plus à un site qui s’appelle *betaseries.com*. Le site permet de mettre un film dans une liste vu, à voir et à ne pas voir. A part cela le site permet de commenter les films et de le noter. Ce site se rapproche de ce que je souhaite faire mais il est à la base, comme son nom l’indique, conçut pour gérer des séries.

L’utilisation de l’API me permet de me concentrer sur la gestion des films et non l’ajout de film qui est un aspect qui ne m’intéresse pas car redondant.

# Analyse fonctionnelle

## Plan du site

Comme on peut le voir ci-dessous on peut faire une recherche de film partout dans le site et la plupart de page on peut y accéder de partout dans le site. Il y a aussi la page connexion qui accessible partout mais seulement si on n’est pas connecté et pareil pour la page inscription.



2 Plan du site

## Maquettes

La page d’accueil, en utilisateur non connecté, avec une pagination. Cette page affiche les films qui ont été ajouté dans des listes, les utilisateurs pourront alors cliquer sur les films pour voir les informations sur le film. On a un bouton connexion et inscription comme dans tout le site si l’on n’est pas connecté. La barre de recherche est aussi accessible depuis tout le site.



3 Maquette page d’accueil déconnecté

Page d’accueil pour utilisateur connecté avec un bouton déconnexion, une icône profil pour aller sur son profil et un lien « Mes listes » qui déroule un menu avec le lien vers ces deux listes. La barre de navigation ne change pas dans les autres pages.



4Maquette page d’accueil connecté

Page d’informations sur un film que l’on a recherché ou cliqué à l’accueil ou dans une liste. Pour l’utilisateur non connecté la différence est que s’il clique sur l’un des boutons qui nécessitent une connexion alors il sera dirigé vers la page connexion. On peut aussi voir les commentaires liés au film, et en ajouter si on est connecté.



5 Maquette page des films

Page des listes avec un bouton pour éditer et un pour supprimer le film de la liste. Si on regarde la liste d’un autre les boutons n’apparaissent pas. On peut cliquer sur le nom du film pour être dirigé vers sa page.



6 Maquette page des listes

Page de connexion



7 Maquette page connexion

Page inscription



8 Maquette page inscription

Page profil d’un utilisateur avec les statistiques de ses listes et un lien pour aller sur celles-ci.



9 Maquette page profil

## Fonctionnalités

L’utilisateur peut :

* Se connecter
* S’inscrire
* Visualiser des informations sur un film qu’il a recherché
* Voir les commentaires d’un film
* Visualiser les listes d’utilisateurs connectés et leur profil
* Visualiser les films qui sont sur l’accueil et en voir les détails sur la page du film
* Naviguer entre chaque page de l’accueil via la pagination

L’utilisateur connecté peut :

* Visualiser des informations sur un film qu’il a recherché et l’ajouter à l’une de ses deux listes
* Commenter un film et voir les commentaires d’un film
* Editer sa propre liste pour supprimer un élément ou le déplacer dans l’autre liste
* Visualiser les listes des utilisateurs connectés
* Visualiser les films qui sont sur l’accueil et y cliquer pour se rediriger vers la page du film
* Naviguer entre chaque page de l’accueil via la pagination
* Se déconnecter

## Logiciels utilisés

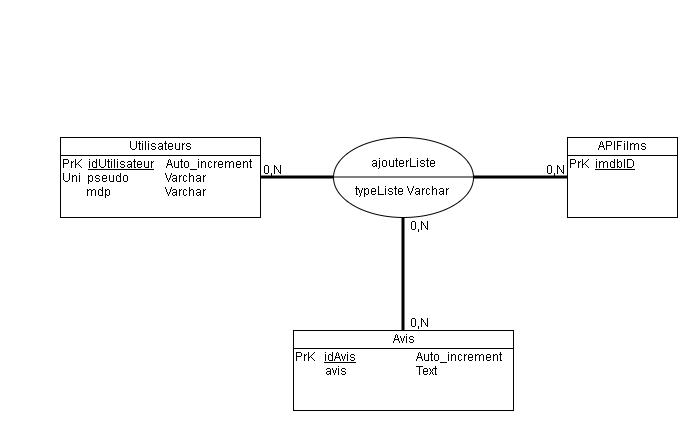
Pour le développement j’ai utilisé NetBeans et EasyPHP 14.1VC9 pour interpréter le PHP. C’est deux logiciels sont ceux qu’on utilise à l’école je suis donc familiarisé avec.

Pour la documentation j’ai utilisé Microsoft Word et Excel pour le texte et le planning et Balsamiq pour les maquettes.

Pour le design de mon site j’ai utilisé un framework CSS qui s’appelle Bootstrap la V4.

# Analyse Organique

## Modèle conceptuel



10 MCD de départ

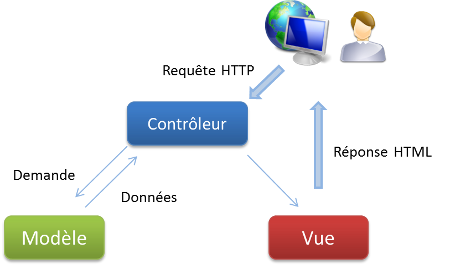


11 MCD actuel

J’ai changé le MCD de départ parce qu’il ne me permettait pas de commenter plus d’une fois un film et il fallait qu’il soit dans l’une des listes. Pour la table APIFilms j’ai rajouté le champ nomFilm pour l’affichage dans les listes sans avoir besoin de faire des requêtes à l’API.

## Fonctionnement détaillé

Le site est fait en HTML et PHP avec un framework CSS, Bootstrap 4 et une API pour récupérer les informations sur les films. J’ai choisi de réaliser mon site en MVC (modèle, vue, contrôleur). Ma page index.php sert de contrôleur, elle fait un « include » de la page à afficher.



12 Schéma MVC source : <https://user.oc-static.com/files/386001_387000/386515.png>

**Contrôleur** : Page index.php, elle fait un « include » de la page à afficher suivant ce qu’elle a reçu.

**Vue** : Autre page (pagefilm.php, accueil.php, liste.php,etc) et les fonctions d’affichages qui ne font pas de traitement mais que de l’affichage.

**Modèle** : Toutes les fonctions qui font un traitement (fonctionsDB.php, fonctionsAPI.php)

**API :** Il s’agit d’OMDb API qui utilise les informations d’imdb, le site avec l’une des plus grosses de base de données de films. Toutes les requêtes sont envoyées à *omdbapi.com* avec mon API key fc17a632et la réponse se fait en json.

## Arborescence

Puisque je fais en MVC l’arborescence ressemble à cela :

* index.php
* Modele
  + fonctionsDB.php
  + fonctionsAPI.php
* Vue
  + accueil.php
  + connexion.php
  + fonctionsAffichage.php
  + pagefilm.php
  + liste.php
  + css
    - maCss.css
  + img
    - loupe.png
    - etc.

Tout ce qui a un point blanc est un fichier ou une page. Les flèches représentent des dossiers.

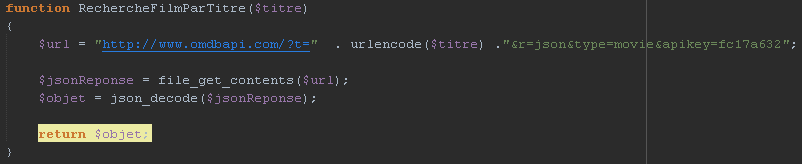
La vue contient tout ce qui est graphique et visuel de mon site et le modèle contient la logique de mon site. On y accède par l’intermédiaire de mon contrôleur qui est ma page index.php.

## OMDb API

OMDb API est une API prenant les données d’imdb, l’API est devenue « payante » en mai 2017 et était gratuite avant cette date. Le paiement de l’API se base sur un financement participatif de 1 $ pour le financement le moins cher jusqu’à 25$ le plus cher (on peut payer plus mais plus aucune contribution au-delà de 25$). J’ai pris celui de 1$ qui me permet un accès basique à l’API et donc une API key personnelle (fc17a632).

Ceci est ma fonction qui me permet de faire une recherche de film par son titre. La fonction prend en paramètre un titre de film à rechercher. Ensuite je vais créer l’url à laquelle je fais la requête en transformant le titre pour qu’il soit compatible dans l’URL. Après on a les paramètres de l’API :

* « t » le nom du film à rechercher, il est obligatoire sauf si on remplace par i qui a besoin de l’imdbid ( l’un des deux doit être présent).
* « r » pour le type des données retournées donc là en json ( par défaut si rien n’est mis c’est en json)
* « type » qu’est-ce qu’il va chercher un film, une série ou même un épisode, là je choisis le film
* « apikey » où l’on indique l’API key, celui-ci fait partit des plus importants car sans lui cela ne marchera pas



13 Fonction API qui recherche un film

J’utilise ensuite file\_get\_contents sur l’url créée avant pour récupérer son contenu. Comme cela me retourne du json il faut que j’utilise json\_decode pour transformer en objet PHP. Et c’est en parcourant cet objet que je prends les informations du film dont j’ai besoin.

J’ai aussi une fonction pour faire la recherche par id du film.

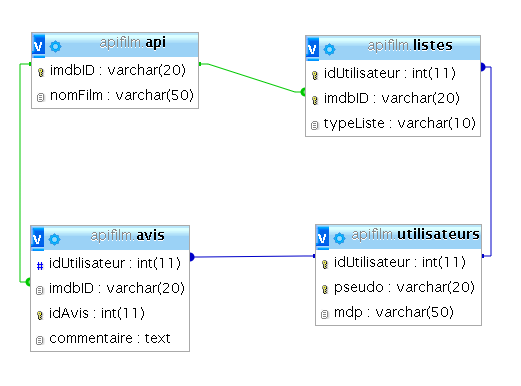
## Sécurité des pages

Un utilisateur non connecté peut visiter les mêmes pages que celui qui est connecté sauf qu’il n’a pas de page personnelle et qu’il ne peut donc pas commenter ou ajouter des films dans des listes. Un utilisateur non connecté ou qui n’est pas sur sa propre page liste ne voit pas des boutons pour changer de liste ni de bouton de suppression.

Par contre en MVC il faut sécuriser chaque page d’affichage pour pas que les utilisateurs puissent y accéder sans que le contrôleur ne donne accès à travers un include. Qui plus est, comme on est en MVC les pages qui ne passent pas par l’index peuvent manquer de certaines informations/variables qui feront que la page génèrera des erreurs. Pour cela je redirige l’utilisateur qui tentera d’aller sur ces pages en tapant le nom dans l’url. J’utilise le fait qu’il ne passe pas par l’index et donc qu’il n’a pas une variable initialisée pour vérifier s’il passe par le contrôleur.

## Base de données

### Modèle physique de la base de données



14 Screenshot du modèle physique

Voici le modèle physique avec mes quatre tables. Ce modèle n’a pas beaucoup changé par rapport au deuxième modèle que j’ai créé, à part l’ajout du nom du film dans la table api pour simplifier l’affichage et éviter de faire des requêtes à l’API tout le temps pour afficher le nom du film dans une liste.

### Description des tables

TABLE UTILISATEURS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Champ | Type | Clé primaire | Auto incrément | Clé étrangère | Description |
| idUtilisateur | **Int(11)** | **X** | **X** |  | **Identifiant de l’utilisateur** |
| pseudo | **Varchar(20)** |  |  |  | **Pseudo de l’utilisateur** |
| mdp | **Varchar(40)** |  |  |  | **Mot de passe en sha1 de l’utilisateur** |

**Table contenant les informations des utilisateurs du site.**

Table API

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Champ | Type | Clé primaire | Auto incrément | Clé étrangère | Description |
| imdbID | **varchar(20)** | **X** |  |  | **Identifiant du film qui vient d’imdb** |
| nomFilm | **Varchar(20)** |  |  |  | **Nom du film en anglais provenant d’imdb** |

**Table contenant l’id du film pour faire une recherche avec l’API et le nom du film pour l’affichage des listes.**

**Table Avis**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Champ | Type | Clé primaire | Auto incrément | Clé étrangère | Description |
| idUtilisateur | **Int(11)** |  |  | **X** | **Identifiant de l’utilisateur** |
| imdbID | **Varchar(20)** |  |  | **X** | **Identifiant du film qui vient d’imdb** |
| idAvis | **Int(11)** | **X** | **X** |  | **Identifiant du commentaire** |
| Commentaire | **Text** |  |  |  | **Le commentaire** |

**Table contenant les commentaires écrits par un utilisateur sur un film.**

**Table listes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Champ | Type | Clé primaire | Auto incrément | Clé étrangère | Description |
| idUtilisateur | **Int(11)** |  |  | **X** | **Identifiant de l’utilisateur** |
| imdbID | **Varchar(20)** |  |  | **X** | **Identifiant du film qui vient d’imdb** |
| typeListe | **Varchar(10)** |  |  |  | **Le nom du type de la liste (vu ou à voir)** |

**Table contenant les listes des utilisateurs.**

### Requêtes SQL

Vérification du login et mot de passe

SELECT pseudo,idUtilisateur FROM utilisateurs

WHERE pseudo = :pseudo and mdp = :mdp

Vérifier si le film est déjà dans la base de données

SELECT \* FROM api

WHERE imdbID = :id

Ajouter un film dans la base de données

INSERT INTO api (imdbID,nomFilm)

VALUES (:idFilm,:nom)

Ajouter un film dans une liste

INSERT INTO listes (typeListe, idUtilisateur, imdbID)

VALUES (:type,:idUtilisateur,:idFilm)

Récupérer une liste de film

SELECT \* FROM listes NATURAL JOIN api

WHERE idUtilisateur = :id and typeListe = :type ORDER BY nomFilm

Supprimer un film d’une liste

DELETE FROM listes

WHERE idUtilisateur = :idUtilisateur and imdbID = :idFilm

Changer un film de liste

UPDATE listes SET typeListe =:NouvType

WHERE idUtilisateur = :idUtilisateur and imdbID = :idFilm

Récupère tous les films qui sont dans une liste

SELECT nomFilm, imdbID, count(imdbID) as nbfilms FROM `listes` NATURAL JOIN api

GROUP BY imdbID order by " . $tri . " " . $ordre . " LIMIT " . (($page-1) \* $limite) . "," . $limite

Compte le nombre de films différents dans les listes

SELECT COUNT(DISTINCT imdbID) as nbFilms FROM listes

Compte le nombre de fois qu’un film apparait dans chacune des listes

SELECT

(SELECT COUNT(typeListe) FROM listes

WHERE typeListe = "aVoir" and imdbID = :id) as nbFilmsAvoir,

(SELECT COUNT(typeListe)FROM listes

WHERE typeListe = "vu" and imdbID = :id) as nbFilmsVu

Compte le nombre de films dans chacune des listes d’un utilisateur

SELECT

(SELECT COUNT(imdbID) FROM listes

WHERE idUtilisateur = :id and typeListe = "vu")as nbVu,

(SELECT COUNT(imdbID) FROM listes

WHERE idUtilisateur = :id and typeListe = "aVoir")as nbAVoir

Récupère les commentaires

SELECT commentaire,pseudo FROM avis NATURAL JOIN utilisateurs

WHERE imdbID = :id

ORDER BY idAvis desc

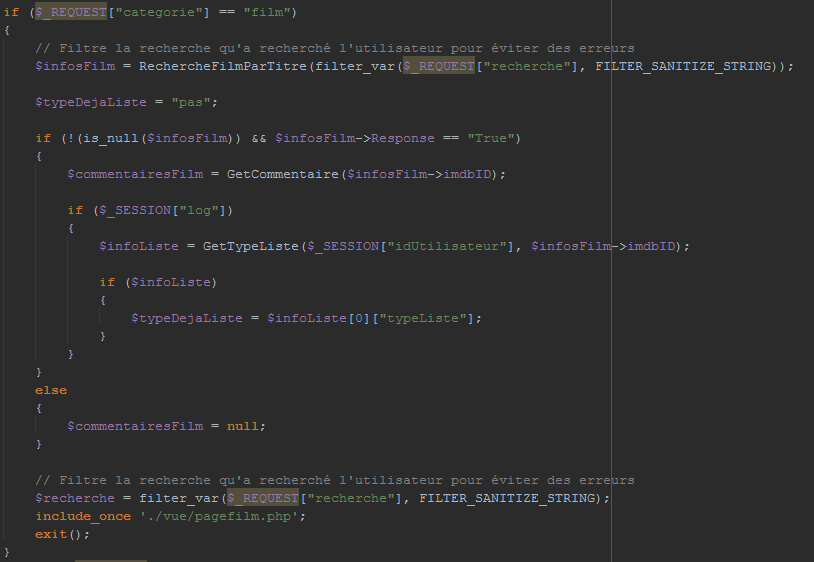
Ajouter un commentaire

INSERT INTO avis (idUtilisateur, imdbID, commentaire)

VALUES (:idUtilisateur,:idFilm,:commentaire)

## Gestion dans le contrôleur

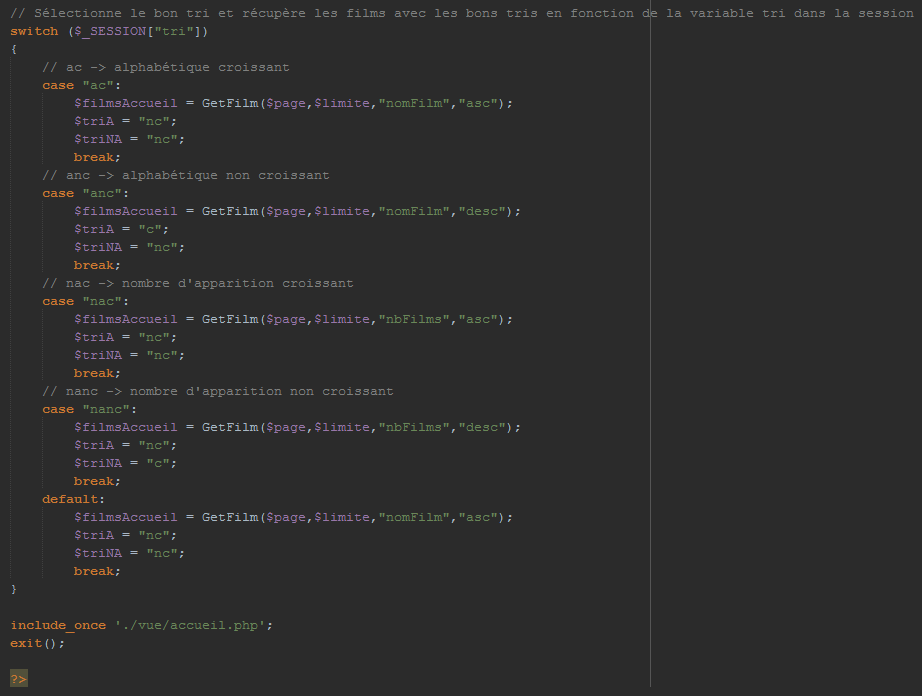
Je vais expliquer comment je gère les redirections des pages que l’utilisateur souhaite accéder. Pour ce faire j’ai pris un screenshot de la gestion de la recherche de film.



15 Screenshot recherche de film

Comme la barre de recherche sert à chercher les films ou les profils je vérifie d’abord qu’il s‘agit d’un film après cela je fais une recherche avec l’api en envoyant le nom, filtrer, que l’utilisateur à entrer. L’api envoi une propriété qui s’appelle Response, qui est true ou false, donc si c’est vrai l’api a trouvé un résultat. Je vais donc après chercher si le film a des commentaires en utilisant son id et une variable prend donc un tableau de commentaires en retour. Je vérifie ensuite si l’utilisateur est connecté pour voir si cet utilisateur a déjà le film dans sa liste. Si c’est le cas je prends le type de la liste et je l’utilise pour désactivé le bouton en rapport avec cette liste. A la fin je prends la recherche effectuée par l’utilisateur pour pouvoir l’afficher dans la barre de recherche. Et enfin je fais un include, donc je vais afficher le contenu de la page « pageFilm.php », pour afficher soit le film avec ses infos soit aucun résultat.

Pour les redirections c’est comme ça pour toutes les pages sauf pour la page d’accueil qui sert de page par défaut. Je regarde donc si une requête a été effectuée, l’envoi d’un formulaire ou un lien cliqué, et si ce n’est pas le cas je passe au suivant. La pour l’exemple la requête reçue c’est la recherche, ensuite j’ai vérifié si c’était un film. Si on n’était pas rentré là-dedans et qu’aucune requête n’étais envoyé je serais renvoyé vers la page d’accueil qui comme elle est la page par défaut n’a pas besoin de requête pour y accéder.



16 Screenshot redirection page accueil

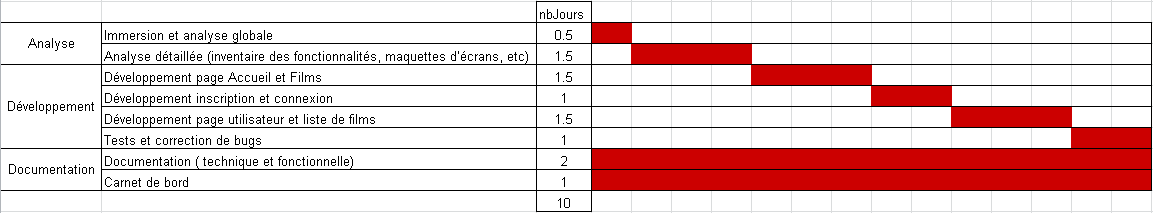
Voilà donc la fin du code qui redirige sur la page d’accueil sans aucune obligation. En haut en prend juste le tri qui doit être effectué sur la liste de film de l’accueil et en récupère ainsi les films triés de cette manière.

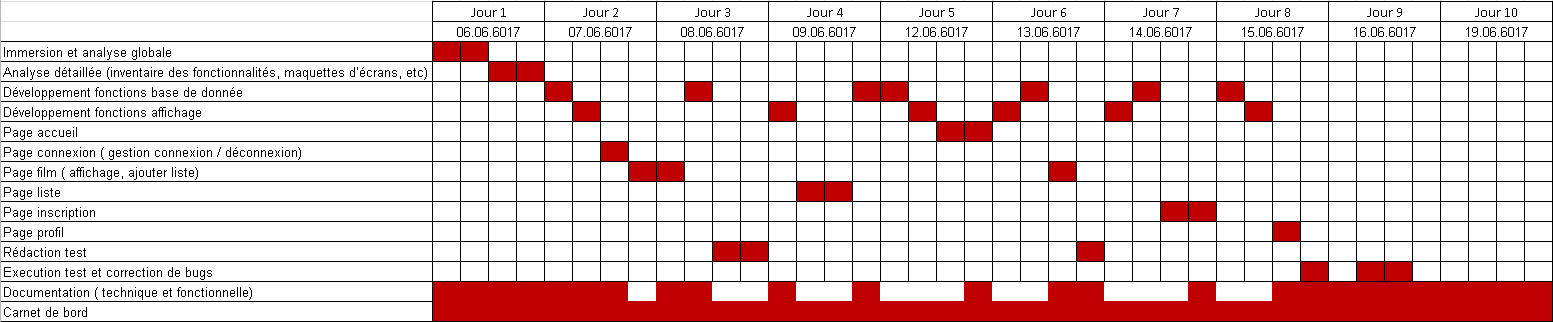
# Tests

|  |  |  |  |
| --- | --- | --- | --- |
| N° | **A faire :** | **Résultat attendu :** | **Résultat obtenu :** |
| T1 | En étant déconnecté je clique sur le bouton se connecter qui se trouve sur la barre de navigation | Je suis redirigé sur la page de connexion | OK |
| T2 | En étant déconnecté je clique sur le bouton s’inscrire qui se trouve sur la barre de navigation | Je suis redirigé sur la page inscription | OK |
| T3 | Cliquer sur « Accueil » qui se trouve sur la barre de navigation | Je suis redirigé sur la page d’accueil | OK |
| T4 | Cliquer sur l’icône qui se trouve sur la barre de navigation | Je suis redirigé sur la page d’accueil | OK |
| T5 | Sélectionner « Film » dans la combobox puis écrire le nom d’un film en anglais dans la barre de recherche puis appuyé sur la loupe | Je suis redirigé sur la page du film que j’ai tapé | OK |
| T6 | En étant déconnecté et sur la page d’un film je clique sur le bouton « Vu » | Je suis redirigé sur la page connexion | OK |
| T7 | En étant déconnecté et sur la page d’un film je clique sur le bouton « A voir » | Je suis redirigé sur la page connexion | OK |
| T8 | Sélectionner « Profil » dans la combobox puis écrire le nom d’un utilisateur dans la barre de recherche puis appuyé sur la loupe | Je suis redirigé sur la page de l’utilisateur que j’ai tapé | OK |
| T10 | Sur la page d’un utilisateur je clique sur « Films Vu » | Je suis redirigé sur la page liste avec les films Vu de cet utilisateur | OK |
| T11 | Sur la page d’un utilisateur je clique sur le lien vers sa liste « A Voir » | Je suis redirigé sur la page liste avec les films A Voir de cet utilisateur | OK |
| T12 | Sur une des deux pages listes d’un utilisateur je clique sur le nom d’un film dans la liste | Je suis redirigé sur la page du film que j’ai cliqué | OK |
| T13 | Sur la page d’accueil je clique sur le nom d’un film | Je suis redirigé sur la page du film que j’ai cliqué | OK |
| T14 | En étant connecté et sur la page d’un film je clique sur « Vu » | Le film est rajouté dans ma liste Vu | OK |
| T15 | En étant connecté et sur la page d’un film je clique sur « A Voir » | Le film est rajouté dans ma liste A Voir | OK |
| T16 | En étant connecté je clique sur « Mes listes » puis sur « A Voir » | Je suis redirigé sur la page A Voir de mes films et tous mes films dans A Voir sont affichés | OK |
| T17 | En étant connecté je clique sur « Mes listes » puis sur « Vu » | Je suis redirigé sur la page Vu de mes films et tous mes films dans Vu sont affichés | OK |
| T18 | En étant connecté je clique sur le bouton de suppression d’un film de la liste qui a l’image d’une croix | Le film est supprimé de ma liste | OK |
| T19 | En étant connecté je clique sur le bouton de modification d’un film de ma liste qui a l’image de deux flèches | Le film a changé de liste | OK |
| T20 | En étant connecté je clique sur le bouton « Déconnecter » | Je suis déconnecté et dirigé vers la page d’accueil | OK |
| T21 | En étant connecté je clique sur l’icône profil | Je suis dirigé vers ma page profil | OK |
| T22 | En étant dans l’une de mes pages listes je clique sur le nom d’un film | Je suis redirigé sur la page du film que j’ai cliqué | OK |
| T23 | En étant déconnecté et sur la page d’un film j’écris un commentaire et je l’envoi | Je suis redirigé sur la page connexion | OK |
| T24 | En étant connecté et sur la page d’un film j’écris un commentaire et je l’envoi | Le commentaire que je viens de taper apparaît dans la section commentaire de ce film | OK |

# Conclusion

## Comparaison planning



17 Planning initial

18 Planning final

Mon planning initial manque un peu d’informations et de détails. Mais au final il représente bien globalement mon planning.

J’ai détaillé plus sur le planning final en séparant chaque page pour que ce soit plus juste. J’ai aussi séparé le développement des fonctions (affichage et base de données) avec le développement des pages (gérer ce que l’on peut faire sur la page). Par exemple pour la page film je n’y pas passé que 3 quarts de jours mais plus si on ajoute le moment où je développais l’affichage des films ou les fonctions liés à la base de données (ajouter film dans une liste, chercher film, commenter film).

En prenant cela en compte presque toutes les pages ont pris un peu plus de temps que prévu. Donc si dès le début j’avais plus détaillé mon planning il aurait été plus facile de planifier un temps plus proche. Mais je suis content d’avoir su prédire dans les grandes ce qui me prendrait le plus de temps.

## Améliorations futures

Au fur et à mesure de l’avancement de mon site j’ai eu des idées qui pourraient améliorer le site. Pour les utilisateurs une auto complétion quand on cherche un film serrai plus user friendly et permettrait de trouver un film si on ne se rappel pas exactement du nom. Il y aurait quelques fonctionnalités à rajouter pour avoir un profil plus complet comme pouvoir changer de mot de passe ou pouvoir avoir une photo de profil. Une dernière fonctionnalité qui pourrait améliorer l’expérience de l’utilisateur serrait la possibilité de supprimer un commentaire.

Les améliorations citées sont des améliorations qui améliorent le confort des utilisateurs. C’est ce qui je trouve serrait à améliorer dans mon site.

## Choix et changements

Une des différences c’est qu’il n’y a pas de tri par note parce que la note n’est pas dans ma base de données. Ce n’est pas vraiment un changement mais la pagination se trouve dans la page d’accueil donc la liste des films déjà ajoutés pas dans la page des listes. Je le précise car dans le cahier des charges il est dit liste mais on peut confondre avec la liste personnelle.

J’ai choisi de créer une page utilisateur alors qu’elle n’est pas obligatoire dans le cahier des charges. La page m’est utile car elle permet, au travers d’un lien se trouvant sur cette page, de consulter les listes d’autre utilisateur.

## Bilan final

Pour finir, ce projet fut intéressant car il m’a permis de mener un projet de A à Z durant un temps limité en suivant un cahier des charges. C’était donc un projet fait dans des conditions de travail proches de la réalité. J’ai pu grâce à ce travail comprendre ce que je devais améliorer et apprendre à me gérer à tous les niveaux.

# Glossaire

|  |  |
| --- | --- |
| **API** | Application Programming Interface, ensemble de methods standardisées, de propriétés, d'évènements, et d'URL servant à interagir avec du contenu web. |
| **PHP** | PHP : Hypertext Preprocessor, langage de programmation de site web. |
| **HTML** | HyperText Markup Language, C’est un langage de balisage permettant d’écrire de l’hypertexte utilisé dans les sites web. |
| **NetBeans** | Il s’agit d’un environnement de développement. |
| **EasyPHP** | Il s’agit d’un interpréteur de script PHP, ayant un serveur web Apache et un serveur de bases de données MySQL. |
| **Balsamiq** | Un site web qui permet de créer des schémas ou des maquettes. |
| **Framework** | Ensemble cohérent de composants éprouvés et réutilisables (bibliothèques, classes) |
| **CSS** | Cascading Style Sheets, langage informatique qui décrit la présentation des documents HTML et XML. |
| **Bootstrap** | Collection d’outils utile à la création du design de site. |
| **URL** | Uniform Resource Locator, chaîne de caractères combinant les informations nécessaires pour indiquer à un logiciel comment accéder à une ressource Internet. |
| **Json** | JavaScript Object Notation, format de données textuelles dérivé de la notation des objets du langage JavaScript. |
| **Combobox** | Champ ayant l’apparence d’un champ de texte mais avec une liste déroulante. |

# Bibliographie

## Source

Pour tout ce qui est image et icône j’ai pris sur ces trois sites :

<https://pixabay.com/fr/loupe-recherche-grossissement-zoom-1083373/>

<http://www.xn--icne-wqa.com/tag-croix-2>

<https://www.shareicon.net/refresh-clockwise-circular-arrows-interface-rotating-two-circle-689139>

L’aide pour le design vient du site de Bootstrap 4, de Bootsnip où l’on retrouve des designs gratuits déjà fait et d’un site répertoriant les classes CSS de Bootsrap :

<https://v4-alpha.getbootstrap.com/getting-started/introduction/>

<https://bootsnipp.com/>

<https://hackerthemes.com/bootstrap-cheatsheet/>

Et pour le code voici la liste des sites que j’ai consultés, php.net car il y a la documentation de php et stackoverflow pour certaines questions que j’ai pu avoir :

<http://php.net/>

<https://stackoverflow.com/>

Je me suis aussi aidé d’autre projet que j’ai fait durant mes trois ans passés dans cette école.

## Tableau de figure

[1 Screenshot du site SensCritique où l’on peut voir une liste créé par un utilisateur 3](#_Toc485567446)

[2 Plan du site 4](#_Toc485567447)

[3 Maquette page d’accueil déconnecté 5](#_Toc485567448)

[4Maquette page d’accueil connecté 5](#_Toc485567449)

[5 Maquette page des films 6](#_Toc485567450)

[6 Maquette page des listes 6](#_Toc485567451)

[7 Maquette page connexion 7](#_Toc485567452)

[8 Maquette page inscription 7](#_Toc485567453)

[9 Maquette page profil 8](#_Toc485567454)

[10 MCD de départ 10](#_Toc485567455)

[11 MCD actuel 10](#_Toc485567456)

[12 Schéma MVC source : https://user.oc-static.com/files/386001\_387000/386515.png 11](#_Toc485567457)

[13 Fonction API qui recherche un film 13](#_Toc485567458)

[14 Screenshot du modèle physique 14](#_Toc485567459)

[15 Screenshot recherche de film 18](#_Toc485567460)

[16 Screenshot redirection page accueil 19](#_Toc485567461)

[17 Planning initial 22](#_Toc485567462)

[18 Planning final 22](file:///F:\CFPT3\TPI\MyMoviesList\Doc\Documentation_technique.docx#_Toc485567463)

# Annexes

Le carnet de bord.

La documentation utilisateur.

Le code source en pdf.