

*Laetitia PHAM*  
*Guillaume RIDEY*  
*Maxence BRUNET*

*Enseignant : Romain DEMANGEON*

---

# UE PC3R

## COMPTE-RENDU

Juin 2020

---



SCIENCE ET TECHNOLOGIE DU LOGICIEL  
SORBONNE UNIVERSITÉ  
ANNÉE 2019/2020

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Manuel d'installation</b>	<b>2</b>
<b>3</b>	<b>Description du client</b>	<b>3</b>
3.1	L'accueil du site . . . . .	3
3.2	Page de description . . . . .	4
3.3	Page de recherche . . . . .	4
3.4	Page de Profil . . . . .	5
3.5	Page d'inscription et d'identification . . . . .	5
<b>4</b>	<b>Description du serveur</b>	<b>6</b>
4.1	Le répertoire: Database . . . . .	6
4.2	Le répertoire: Servlet . . . . .	6
4.3	Le répertoire: Services . . . . .	7
4.4	Le répertoire: tools . . . . .	7
4.5	Le répertoire: test . . . . .	7
<b>5</b>	<b>L'interaction avec l'utilisateur</b>	<b>8</b>
5.1	Fonctionnement global du client . . . . .	8
5.2	Les boutons favoris . . . . .	8
5.3	Redirection titre et image . . . . .	8
5.4	Redirection logo . . . . .	8
5.5	Gestion des messages . . . . .	8
5.6	Gestion des commentaires . . . . .	8
<b>6</b>	<b>Les Points forts</b>	<b>8</b>
6.1	Bootstrap . . . . .	8
6.2	Mongodb . . . . .	9
6.3	Commentaires dynamiques . . . . .	9
6.4	Copie locale des favoris . . . . .	9
<b>7</b>	<b>Les éventuelles difficultés rencontrés</b>	<b>9</b>
7.1	Page de recherche et chargement . . . . .	9
7.2	Montage et démontage d'un composant . . . . .	9
7.3	Changement des boutons favoris . . . . .	9
7.4	Données JSON de l'API externe . . . . .	10
<b>8</b>	<b>Conclusion</b>	<b>10</b>

# 1 Introduction

L'objectif de ce projet est de proposer un site internet dynamique, en prenant en compte l'utilisation d'une API externe régulièrement mise à jour.

L'API **The Movie DB** semblait être le choix le plus intéressant, car elle contient une large base de données sur les séries et les films. Les films et les séries étant des éléments régulièrement mises à jour (en terme de sortie ou d'actualité), cela nous permet donc d'obtenir un site qui est actualisé quotidiennement.

Notre site internet est un calendrier des films sortis depuis 2020 (ceux proposés par l'API externe), ainsi qu'un calendrier des sorties des séries des sept derniers jours. Nous mettons en place plusieurs interactions avec l'utilisateur en lui permettant d'effectuer des actions telles que la recherche, ou s'il est connecté, l'ajout de favoris ou bien de commenter des films et des séries.

Dans la suite de ce rapport, nous allons expliquer les outils à installer pour le bon fonctionnement du site. Puis nous allons décrire l'ensemble du client et une partie du serveur (le serveur ayant été décrit lors d'un TME). Finalement, nous verrons les points forts et les difficultés rencontrées durant ce projet.

## 2 Manuel d'installation

En ce qui concerne l'installation du linux, on a besoin de :

**MySQL :** L'installation de mySQL est importante puisqu'elle garde en mémoire les comptes des utilisateurs du site internet (Nom de compte et mot de passe). Il faut également **ajouter notre base de données "MochiCine.sql"** sur le site "Phpmyadmin". Notre base de données est **présente à la racine du dossier du projet** (Le mot de passe et l'utilisateur sont tout deux mis par défaut à "root" dans le fichier *database/DBStatic.java*).

<https://www.digitalocean.com/community/tutorials/comment-installer-mysql-sur-ubuntu-18-04-fr>

**Mongodb :** Mongodb garde en mémoire les commentaires et les favoris d'un utilisateur.

<https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>

**React JS :** Pour installer React JS :

-**node js** : `sudo apt install nodejs`

-**npm** : `npm install --global npm`

**TomCat v9.0 :** Pour faire le lien entre notre serveur et le client.

Pour **lancer le projet**, il faut lancer le serveur Tomcat (via eclipse ou avec le navigateur), lancer mongodb avec la commande "service mongod start", et enfin à la racine du projet, lancer la commande "sudo npm start".

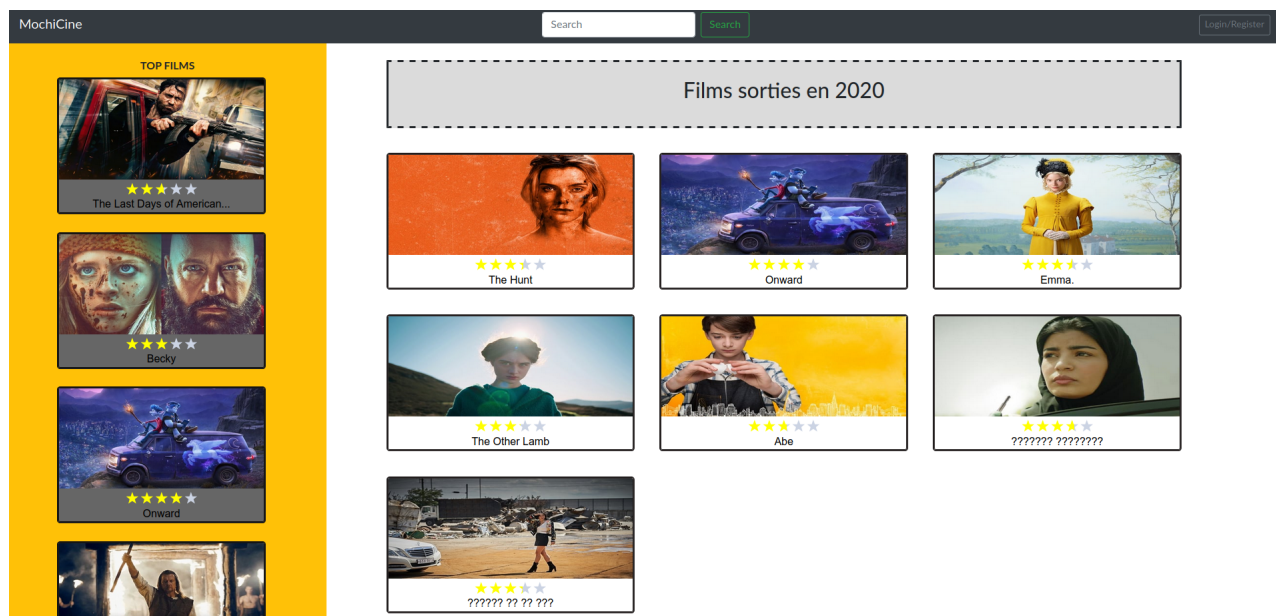
### 3 Description du client

Pour notre site internet, nous avons décidé de mettre en place un site mono-page. En effet, cela nous permet de rendre nos pages plus dynamiques et d'utiliser la bibliothèque *ReactJS* que nous avons déjà étudié l'année précédente.

Le site comptabilise un total de cinq pages. Toutes les pages sont divisées en deux parties, la barre de menu en haut et le reste de la page en bas.

La barre de menu est identique pour toutes les pages. Elle comporte au centre une barre de recherche et un bouton pour lancer cette recherche. À gauche de cela se situe le nom de notre site, *MochiCine*, qui permet à l'utilisateur de revenir à l'accueil. À droite un bouton permettant de se connecter/s'inscrire, ou de se déconnecter si l'utilisateur était déjà connecté.

#### 3.1 L'accueil du site



L'accueil du site est constitué d'une partie qui regroupe les films sortis en 2020 puis les séries sorties ces sept derniers jour. Mais également, d'une seconde partie constituée des tendances du moment en termes de séries et de films.

Pour chacun d'entre eux, on a une image, une note attribuée par la communauté du site qui fournit l'API externe, ainsi que le titre. Certains titres apparaissent avec des "?", en effet cela signifie que le titre utilise un alphabet différent du latin, ainsi il n'a pas été transcrit par l'API externe.

Lorsqu'un utilisateur est connecté, il peut également voir apparaître l'icône d'ajout aux favoris pour chaque film ou série de l'accueil. Lorsqu'on ajoute un film ou une série en favoris, elle apparaît directement sur la page de profil de l'utilisateur.

L'utilisateur peut appuyer sur l'image ou le titre d'un film, pour aller sur la page de description de ce dernier.

## 3.2 Page de description


MochiCine

Search

Search

Profile

Logout



### Avengers: Infinity War

25 avril 2018 61.241 film

As the Avengers and their allies have continued to protect the world from threats too large for any one hero to handle, a new danger has emerged from the cosmic shadows: Thanos. A despot of intergalactic infamy, his goal is to collect all six Infinity Stones, artifacts of unimaginable power, and use them to inflict his twisted will on all of reality. Everything the Avengers have fought for has led up to this moment - the fate of Earth and existence itself has never been more uncertain.

Add Favourites

#### Commentaires

Add a Comment...

Comment

alice

Trop cool

2020/06/07 10:24:47

Realisateur(s) :

Genre(s) :

Status :

Adventure

Action

Science Fiction

Released

La page de description est constituée d'une image, d'un titre, d'une synopsis, de la date de sortie et du type (film ou série en décrivant le nombre de saisons).

En dessous de l'image, il y a le nom des réalisateurs (lorsqu'ils sont décrits via l'API externe), le genre et le statut, c'est-à-dire déjà sortie ou non.

Par ailleurs, une section commentaire est également présente. En effet, les commentaires sont visibles par tout le monde mais seuls les utilisateurs connectés peuvent écrire un commentaire ou supprimer leur propre commentaire. De plus, chaque commentaire est composé de la date et l'heure de publication.

Comme pour la page d'accueil, on peut également ajouter le film ou la série en favoris, cela sera directement affiché sur la page de profil de l'utilisateur.

## 3.3 Page de recherche


MochiCine

Search

Search

Profile

Logout




### Jurassic World

6 juin 2015 31.649 movie

Twenty-two years after the events of Jurassic Park, Isla Nublar now features a fully functioning dinosaur theme park, Jurassic World, as originally envisioned by John Hammond.

Add Favourites



### Jurassic Galaxy

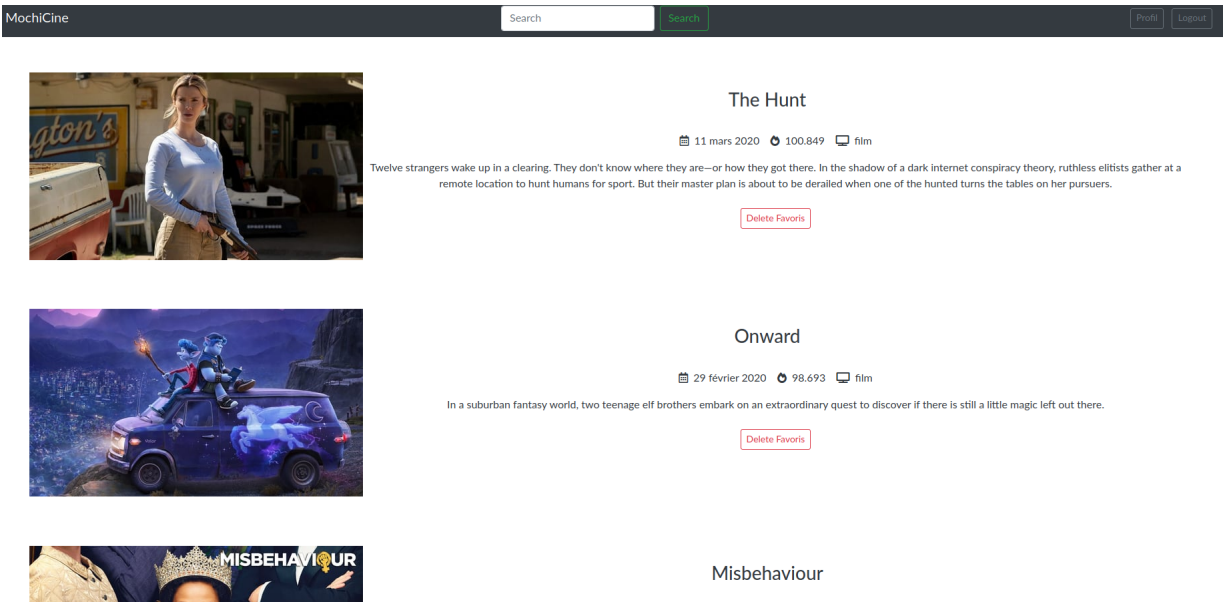
20 août 2018 22.421 movie

In the near future, a ship of space explorers crash land on an unknown planet. They're soon met with some of their worst fears as they discover the planet is inhabited by monstrous dinosaurs.

Add Favourites

On accède à la page de recherche en tapant le nom d'une série ou d'un film dans la barre de recherche. Ainsi, on obtient une description brève de chacun des résultats de la recherche. Ici encore, un utilisateur connecté peut ajouter un film ou une série en favoris. En allant sur le nom d'un des résultats, cela permet de se rendre sur la page de description de ce dernier.

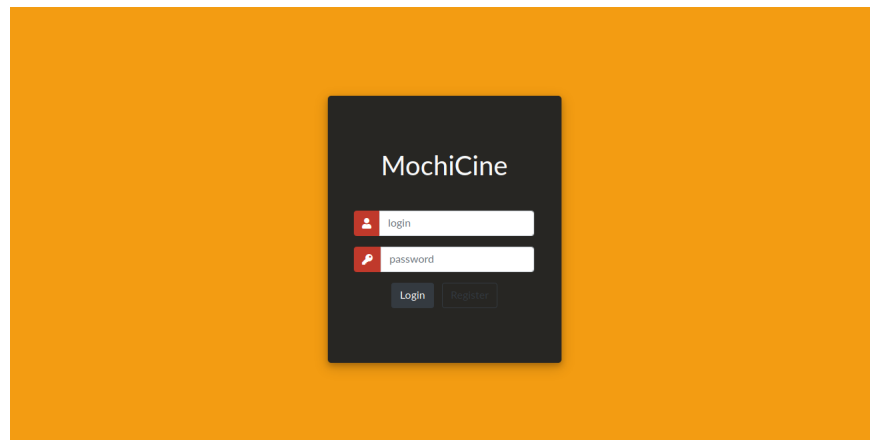
### 3.4 Page de Profil



La page de profil ressemble beaucoup à la page de recherche, mais plutôt que d'afficher les films d'une recherche elle affiche tout les favoris de l'utilisateur.

Il y a en bas de chaque favori un bouton *Delete Favors* pour supprimer ce favori de la liste de l'utilisateur. L'utilisateur peut aussi cliquer sur l'image ou le titre d'un favori pour se rendre sur sa page de description.

### 3.5 Page d'inscription et d'identification



Cette page permet de se connecter grâce au bouton *"Login"*, ou bien de créer un nouveau compte utilisateur avec le bouton *"Register"*. De plus, en appuyant sur le nom du site, *MochiCine*, cela permet de revenir à l'accueil.

## 4 Description du serveur

En ce qui concerne l'architecture de notre serveur, il est constitué de cinq packages.

### 4.1 Le répertoire: Database

Cela correspond à la connexion réalisée entre notre serveur et nos bases de données. D'un côté on crée une connexion vers mySQL avec la méthode "getMySQLConnection()" et de l'autre une connexion vers MongoDB avec "getMongoConnection()".

### 4.2 Le répertoire: Servlet

Ce répertoire contient les servlets qui effectuent la correspondance entre le client et le serveur. Lorsqu'un client réalise une action, une requête serveur est envoyée via l'un des servlets correspondant. Nous avons un servlet par action :

**AddComment** : Lorsqu'un utilisateur souhaite rédiger un commentaire sur le post d'un message d'un film ou d'une série. La requête prend 3 paramètres: Un login, l'identifiant du message, un commentaire.

**AddFavoris** : Lorsqu'un utilisateur souhaite ajouter un film ou une série en favoris. Il prend 3 paramètres: un login, l'identifiant du film ou de la série, un booléen qui définit si c'est une série ou non.

**AddMessage** : Lorsqu'un utilisateur souhaite rédiger un message sur un film ou une série (pour donner un avis). La requête prend 3 paramètres: Un login, un titre, le texte du message.

**DeleteComment** : Lorsqu'un utilisateur souhaite supprimer un commentaire. La requête prend 2 paramètres: L'identifiant du message et l'identifiant du commentaire.

**DeleteFavoris** : Lorsqu'un utilisateur souhaite supprimer un film ou une série de ses favoris. La requête prend 3 paramètres: Un login, l'identifiant du favori, un booléen qui définit si c'est une série ou non.

**DeleteMessage** : Lorsqu'un utilisateur souhaite supprimer un message d'un film ou une série. La requête prend un seul paramètre : L'identifiant du message.

**FilmsOnAir** : Permet de récupérer la liste des films sortis depuis 2020 (Ceux disponibles via l'API externe). La requête prend un paramètre : La clé fournie par l'API externe du site.

**GetDescription** : Récupère la description d'un film ou d'une série. La requête prend 2 paramètres: L'identifiant du film ou de la série ainsi qu'un booléen qui définit si c'est une série ou non.

**ListFavoris** : Permet de récupérer la liste des favoris pour un utilisateur donné. La requête prend un paramètre : Un login.

**ListMessages** : Permet de récupérer la liste des messages d'un film ou d'une série. La requête prend un paramètre : Un titre d'un film ou d'une série.

**Login** : Pour permettre à un utilisateur existant dans la base de données de se connecter. La requête prend 2 paramètres : Un login et un mdp.

**Logout** : Pour déconnecter un utilisateur connecté. La requête prend en paramètre le login de l'utilisateur.

**Register** : Permet de créer un compte. La requête prend 2 paramètres: Un login et mot de passe.

**Search** : Permet d'effectuer une recherche par mots clés. La requête prend 2 paramètres : La clef fournis par l'API externe du site et un mot clé.

**SerieOfTheWeek** : Permet de récupérer la liste des séries sur les 7 derniers jours. La requête prend en paramètre la clef fournis par l'API externe du site.

**Tendances** : Permet de récupérer la liste des tendances en terme de série et de film. La requête prend 3 paramètres: La clef fournis par l'API externe du site , un type (si c'est un film ou une série), le nombre de films/série que l'on souhaite récupérer.

### 4.3 Le répertoire: Services

Les services concernent les classes Java qui traitent les requêtes liées aux Servlets. On catégorise chaque requêtes. On comptabilise 6 services :

**APIService** : Qui traite les requêtes liées à l'API externe.

**Authentification** : Qui traite les requêtes liées à la connexion ou la déconnexion d'un utilisateur.

**Messages** : Qui traite les requêtes liées aux publications de messages sur le descriptif d'un film ou d'une série, ou leur suppression si ceux-ci ont été publié par l'utilisateur.

**Commentaires** : Qui traite les requêtes liées à l'ajout ou la suppression de commentaires d'un message d'un utilisateur.

**Enregistrement** : Qui traite les requêtes liées à la création d'un compte utilisateur.

**Favoris** : Qui traite les requêtes liées à l'ajout ou la suppression de favoris d'un utilisateur.

Nos services vérifient au préalable que les paramètres données soient correctes, ainsi que certain pré-requis comme la vérification de l'identité de l'utilisateur. Ensuite, ils appelleront la méthode nécessaire au traitement de la requête en utilisant les classes dans le package "tools". En effet, celles-ci feront les différents appels aux bases de données concernées.

### 4.4 Le répertoire: tools

Ce répertoire contient les différentes méthodes effectuant des requêtes sur nos bases de données. Il n'est pas nécessaire de décrire les différentes classes dans cette partie: pour une meilleur compréhension, l'ensemble du code est détaillé et commenté.

### 4.5 Le répertoire: test

Nous avons aussi réalisé une série de tests unitaires afin de vérifier le bon fonctionnement des méthodes de Tools avant leur déploiement coté client.

Nous avons ainsi dans *TestMessage*, les testes d'ajout et de suppression de commentaires et de messages. Puis, nous avons le test, *TestUsers*, qui vérifie le bon fonctionnement de la connexion et de la déconnexion d'un utilisateur. Dans *TestFavoris*, nous testons l'ajout et la suppression d'un favori. Enfin, dans *TestRegister*, nous testons la création d'un nouveau profil.

Pour facilité la compréhension du lecteur sur le code présent dans le client, nous allons expliquer l'interaction du l'utilisateur sur nos différentes pages.



## 5 L'interaction avec l'utilisateur

### 5.1 Fonctionnement global du client

Pour naviguer de page en page sur notre site internet, on modifie l'état de la page courante dans le fichier **MainPage.js**. La fonction de render affiche la page dont l'état courant est spécifié. Lorsque l'état de page courante est modifié (cad lorsque l'on click sur un bouton spécifique), automatiquement la page change.

### 5.2 Les boutons favoris

Lorsqu'un utilisateur clique sur un bouton favoris, on appelle une méthode "handleAddFav" (pour un ajout) ou "handleDeleteFav" (pour une suppression). Cette méthode effectue un appel au servlet addFavoris/DeleteFavoris du serveur pour effectuer un ajout ou une suppression de favoris. Lorsqu'un film ou une série est ajouté, on change le logo affiché et réciproquement lorsqu'on supprime un favori.

### 5.3 Redirection titre et image

Dans les pages de profils (**Profil.js**), d'accueil (**Accueil.js**) et de recherche (**SearchPage.js**), il est possible d'accéder à la page de description d'un film ou d'une série en cliquant sur son titre ou sur son image. Lorsqu'un clic est effectué sur le titre ou l'image, la méthode "HandleDescriptionPage" est appelée. Cette méthode effectue une requête au servlet "GetDescription" (cf 4.2) et redirige vers la page de description.

### 5.4 Redirection logo

Sur la barre de navigation (**NavBar.js**) le logo "MochiCine" permet de revenir sur la page d'accueil. Lorsque l'on clique sur le logo, on appelle la méthode "sendaccueil" qui modifie la page courante à afficher.

### 5.5 Gestion des messages

Dans la page de description d'un film ou d'une série il est possible d'ajouter un message. Pour ajouter un message il faut l'écrire dans la zone de saisie puis valider en cliquant sur *comment*. Lorsque l'utilisateur clique sur *comment* la méthode "sendMessage" effectue un appel au servlet "AddMessage" du message pour ajouter un message.

Une fois qu'un message est validé, il apparaît en bas de la page de description. L'utilisateur qui a créé ce message peut désormais le supprimer en cliquant sur le "symbole de poubelle" se situant à droite de son message. Lors de la suppression, on appelle le servlet "DeleteMessage" pour le supprimer de la base de données MongoDB. Un utilisateur peut commenter un message posté par un autre utilisateur ou le sien.

### 5.6 Gestion des commentaires

Toujours sur la page de description d'un film ou d'une série, il est possible d'ajouter un commentaire sur un message posté d'un utilisateur. Il suffit d'appuyer sur la petite bulle en bas à gauche du commentaire. Lorsque l'on poste un commentaire, on effectue un appel au servlet "addComment" qui ajoute le commentaire dans notre base de données MongoDB (Et réciproquement pour la suppression d'un commentaire).

## 6 Les Points forts

Notre projet possède plusieurs points forts importants à souligner.

### 6.1 Bootstrap

L'utilisation de Bootstrap nous permet d'avoir un site réactif en terme d'affichage, c'est-à-dire que nos composants s'adaptent en fonction de la taille de la fenêtre du navigateur.

## 6.2 MongoDB

L'utilisation de MongoDB nous permet de traiter facilement et rapidement nos données sur des éléments tels que des messages ou des commentaires.

## 6.3 Commentaires dynamiques

Les films et les séries possèdent un système de commentaire. Ce système permet aux utilisateurs de pouvoir donner leur avis sur le film ou la série qui les intéresse.

Dès qu'un utilisateur écrit un commentaire, il est publié puis enregistré sur le serveur via MongoDB. Un autre utilisateur peut également commenter le commentaire d'un autre utilisateur. Ce qui permet à deux utilisateurs d'avoir une discussion dans l'espace commentaire.

## 6.4 Copie locale des favoris

Notre site garde une copie locale des favoris de l'utilisateur, en effet, cela permet de choisir lors de l'affichage des films ou séries si le bouton doit être celui de l'ajout ou celui de la suppression. Ainsi, il n'est pas nécessaire de faire la requête au serveur pour le savoir, ce qui favorise la performance et rapidité de notre site.

# 7 Les éventuelles difficultés rencontrées

Nous avons rencontré plusieurs difficultés que nous allons expliquer ci-dessous.

## 7.1 Page de recherche et chargement

Lorsque nous sommes sur la page des résultats d'une recherche, on laisse la possibilité à l'utilisateur d'effectuer de nouveau une recherche. Cependant cela peut causer des erreurs lors du chargement de la nouvelle recherche.

En effet, l'appel à l'API externe depuis le serveur peut prendre beaucoup de temps, car le site qui propose l'API n'est pas complet. Ainsi nous sommes parfois obligés d'effectuer plusieurs requêtes en même temps, ce qui prend un certain temps, or ce temps est parfois plus long que le chargement de la page elle-même. Les éléments à afficher sont alors à la valeur "nul" ou "indéfini" dans le *json* reçu.

Pour palier à cette éventualité, on peut parfois voir apparaître très rapidement "Chargement en cours" (au lieu d'une page blanche). Cela signifie que les requêtes sont en cours de traitement. Ainsi grâce à la mise en place de cette technique, lorsque les requêtes seront effectuées, la page se rafraîchira pour les afficher.

## 7.2 Montage et démontage d'un composant

Cette difficulté est en lien avec la précédente. En effet, lorsque l'on est situé sur la page de recherche et qu'on effectue une recherche, on démonte puis remonte ce composant. Parfois le montage n'est pas assez rapide et on peut se retrouver avec une erreur "Component is unmount", notamment lors de la mise à jour d'un composant.

Par exemple, lorsqu'on effectue une recherche sur la page de recherche, celle-ci n'est pas rechargée, mais elle est mise à jour. Comme les requêtes à l'API externe sont parfois plus rapides que le montage/démontage du composant cela provoque une erreur. Ce problème a été résolu en vérifiant à plusieurs moments que le composant soit monté avant de faire une mise à jour.

## 7.3 Changement des boutons favoris

Une des premières difficultés que nous avons rencontrées, a été le changement d'action d'un bouton lorsqu'on appuie dessus.

Par exemple le bouton permettant à un utilisateur d'ajouter un film à ses favoris. Une fois le film ou la série ajoutée, il doit directement se transformer en un bouton de suppression du favori. Puisque ces deux actions appellent des méthodes différentes, il a fallu faire bien attention à ce que le changement d'action soit bien mis en place.

## 7.4 Données JSON de l'API externe

Le *JSON* récupéré de l'API externe après une requête n'est pas tout le temps complet. Par ailleurs, les attributs diffèrent d'un film d'une série, ou bien certaines informations peuvent être indiquées dans un champ puis dans un autre. Ainsi, nous avons dû faire pas mal de condition et de "ou" logique pour être sûr que l'information soit recueillie.

De plus, lorsque l'utilisateur veut accéder à la description d'un film ou d'une série dans l'accueil ou la page de recherche, une nouvelle requête doit être envoyée au serveur pour récupérer toutes les données. En effet, contrairement à la page de profil qui réutilise le *JSON* de la requête des favoris pour afficher la description, ces pages ayant fait leurs requêtes spécifiques ne peuvent pas être réutilisé par la page de description.

## 8 Conclusion

Nous avons mis en place un site internet respectant le cahier des charges. Le site interne est dynamique: Les films et les séries sont actualisés tous les jours et l'ajout de commentaires, de messages ou de favoris se fait en temps réel. Nos bases de données sont donc actualisés régulièrement. La mise en page du client, les requêtes serveur et la création du server sont des éléments qui nous ont prit beaucoup de temps: on sécurise au maximum les données utilisateurs, on a commenté l'ensemble du code (côté serveur et client), vérifier que le site fonctionne sur différents navigateurs etc.. Le site internet que nous avons mis en place tme 6 à subir quelques changements. Certains paramètres tel que l'affichage des séries de la semaine n'est plus d'actualité: Le lundi très tôt, il n'existe pas de sortie de nouvelles série. La page d'accueil se retrouvait vide. Malgré différents problèmes lié à l'API externe (ex: cf 7.4), elle s'est avérée bien construite et pratique à utiliser. Nous sommes satisfait du résultat de notre projet. Si nous avions eu plus de temps nous aurions amélioré l'aspect esthétique du site et ajouté une recherche avancée (par auteur, par date etc..)