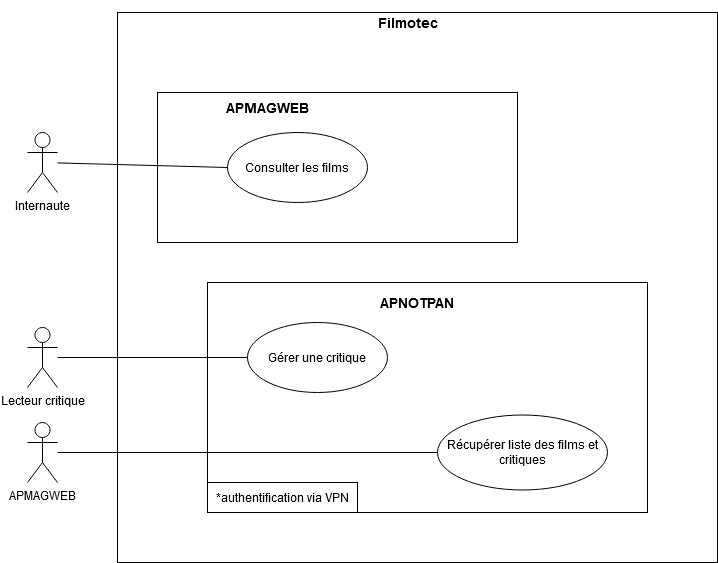
Pisano Vincent Vasseur William Morel Emilien

**Rapport :**

**Introduction :**

Le projet Filmotec consiste principalement à la création de l’application APNOTPAN. Celle-ci doit permettre d’évaluer des films en leur attribuant une note destinée à la critique et un avis. APNOTPAN est une application interne qui sera accessible pour les utilisateurs du panel via un VPN d’entreprise. De plus, un certificat personnel permettra de faciliter les authentifications et les autorisations. L’application utilisera l’API TMDB comme source des données afin de référencer les films sélectionnés.

**Cas d’utilisation UML :**



**Choix de VueJS à la place de NodeJS:**

Nous avons choisi VueJS comme technique d’architecture car l’équipe est plus à l’aise et opérationnelle sur ce Framework que sur NodeJS.

**Présentation de l’API REST :**

L’api rest que nous avons créé à 3 méthodes principales :

* La première méthode « requestUser » prend comme paramètres le nom d’utilisateur et le mot de passe. Si ces deux paramètres sont retrouvés dans la base de données alors la méthode retourne dans un json un tableau avec l’iduser unique, le nom d’utilisateur(name) et le mot de passe(password).
* La seconde méthode « insertOpinion » qui prend comme paramètres l’avis(review), la note(rate), l’iduser et l’idmovie. Cette méthode réalise une insertion dans la table « opinion » de la critique réalisée par l’utilisateur.
* La troisième méthode « getOpinions » prend comme seul paramètre l’iduser. Elle retourne toutes les critiques faites par cet utilisateur dans un json ou chaque critique est dans un objet contenant l’idmovie, la note et le message.

**Présentation de l’API APMAGWEB :**

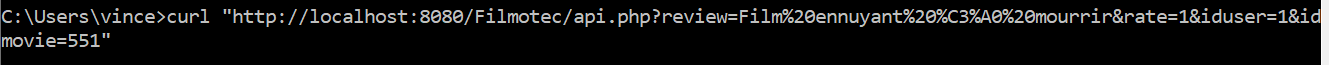
La seule méthode de l’api est « getListMensuel » qui retourne la note maximum et minimum ainsi que le commentaire pour tous les films critiqués.

**Présentation CURL de l’API REST :**

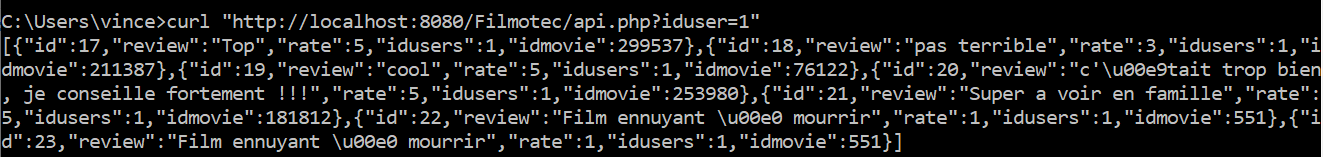
« requestUser » :



« insertOpinion » :

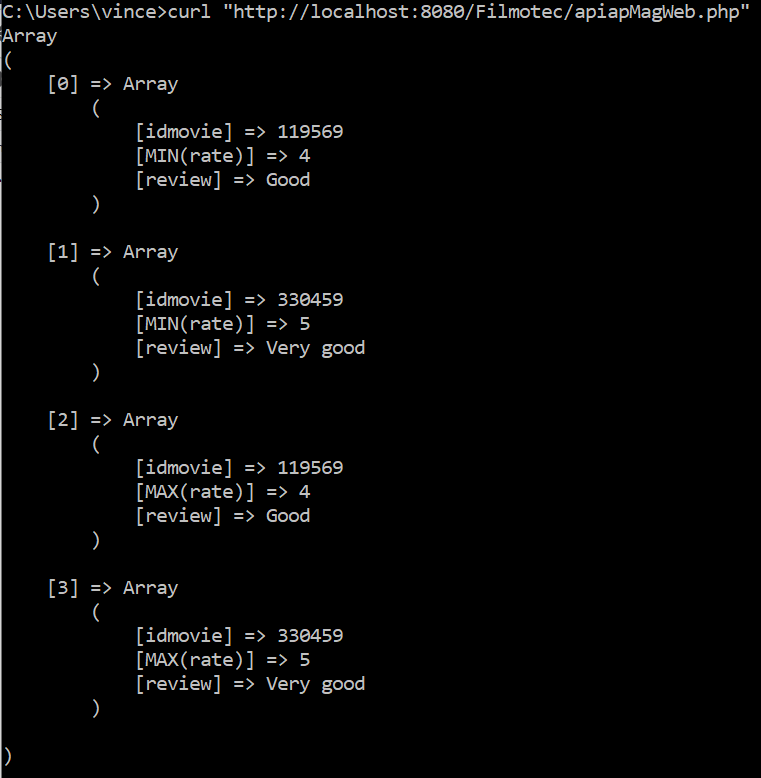


« getOpinions » :



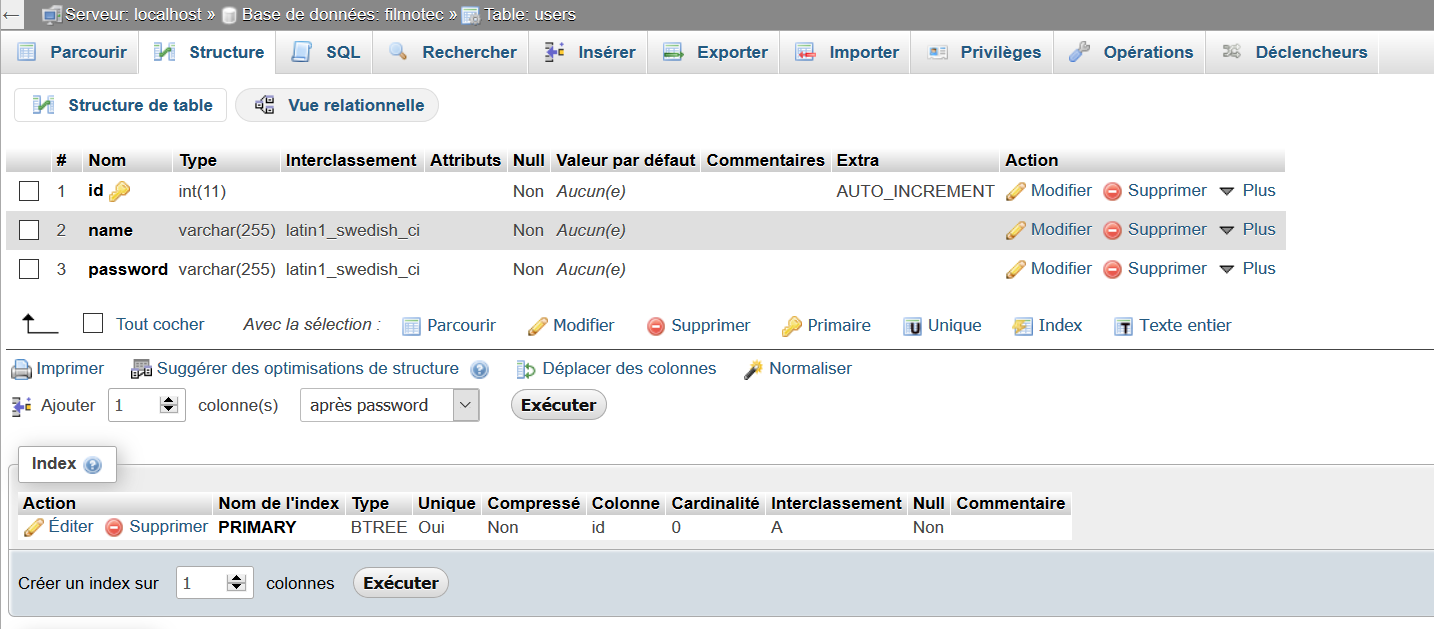
**Présentation CURL de l’API APMAGWEB :**

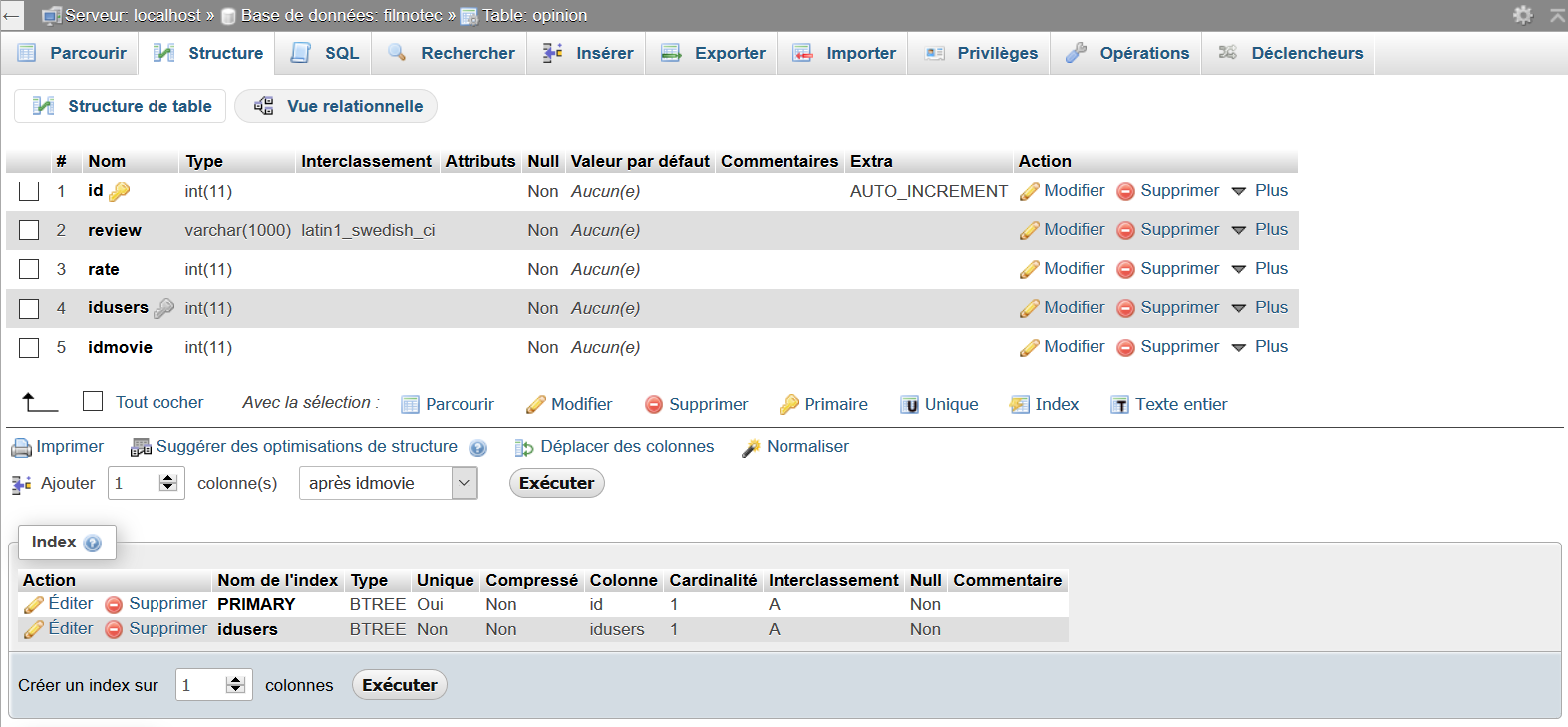
« getListMensuel » :



**Présentation de la base de données :**

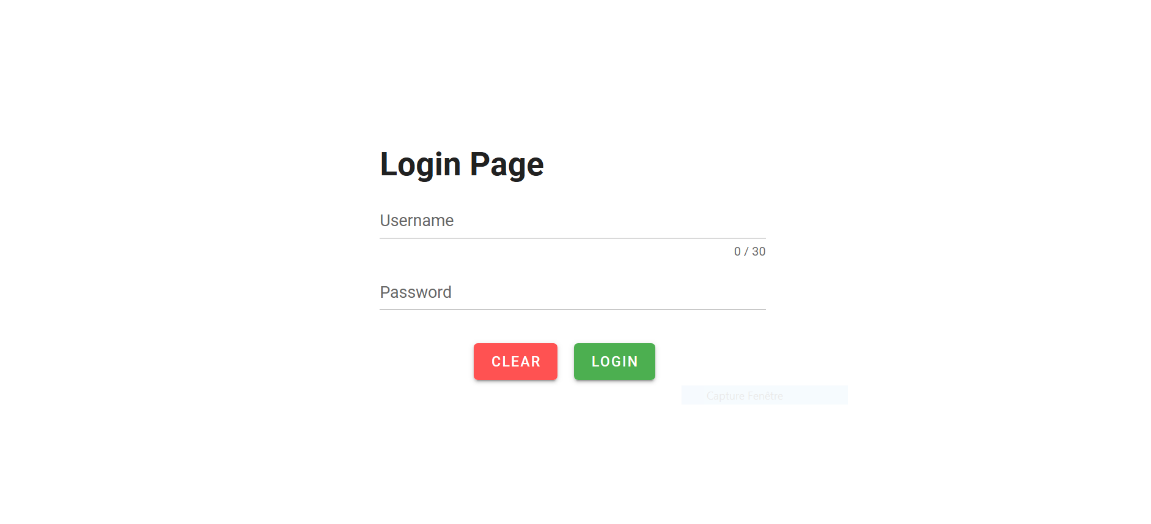
La base de données que nous avons créée se nomme « Filmotec », elle possède 2 tables. La première user qui a 3 champs : un id, un name et un password.



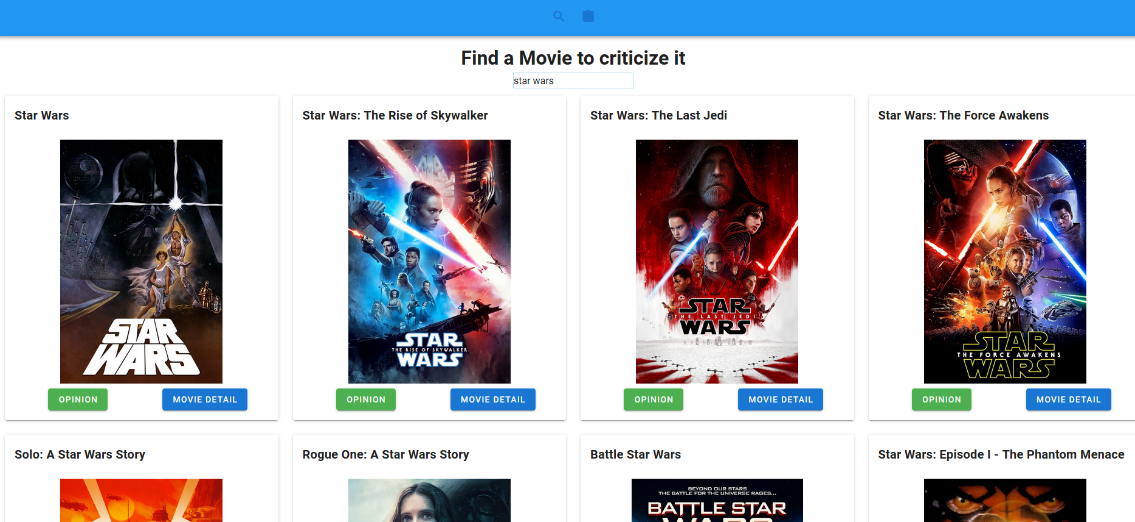
La deuxième table est la table « opinion » avec 5 champs : id, review, rate, idusers et idmovie. L’iduser est une clé étrangère liée à l’id de la table user. 

**Présentation du site APNOTPAN :**

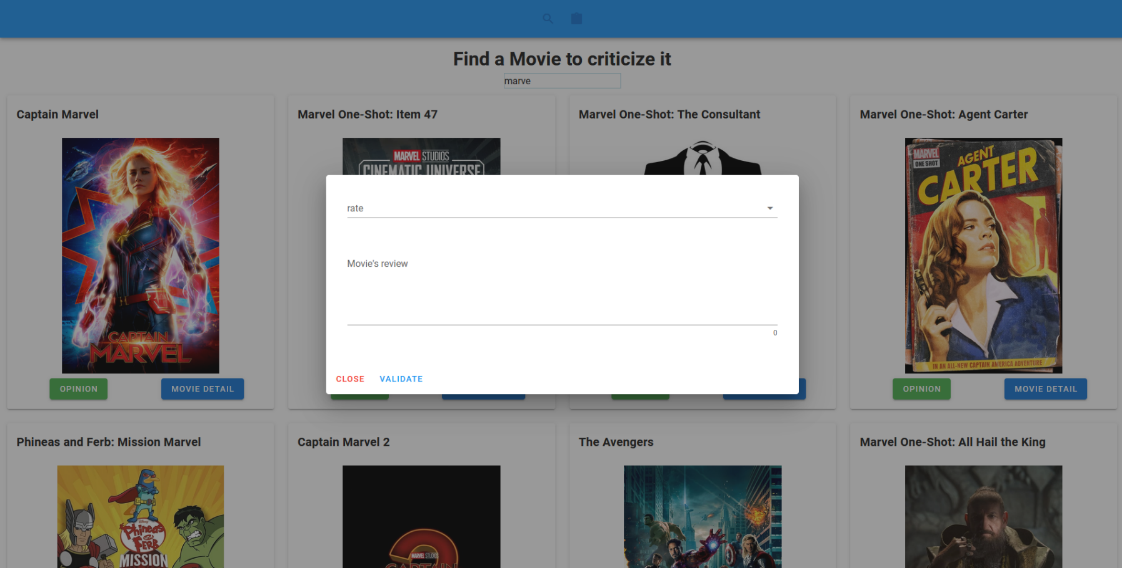
Une page de connexion classique qui permet d’accéder à l’application avec un nom d’utilisateur et un mot de passe.



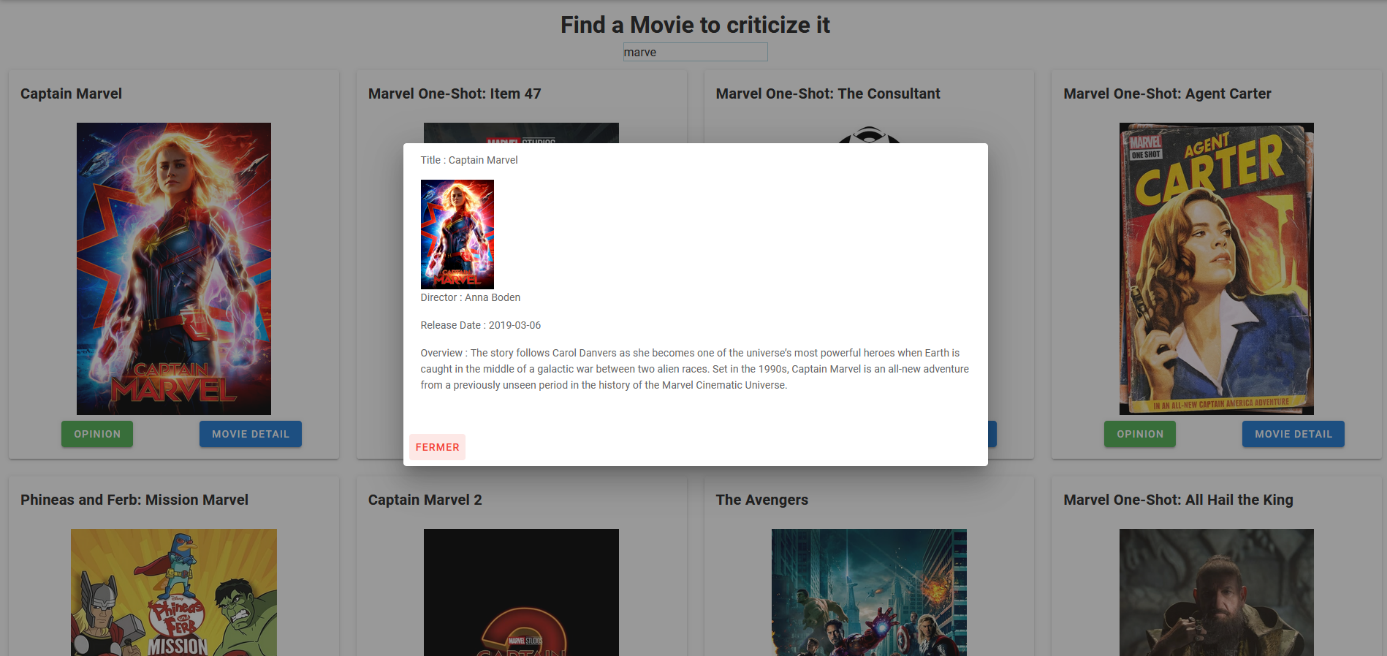
Une page de recherche permettant de trouver les films que l’on veut critiquer grâce à un champ de recherche.



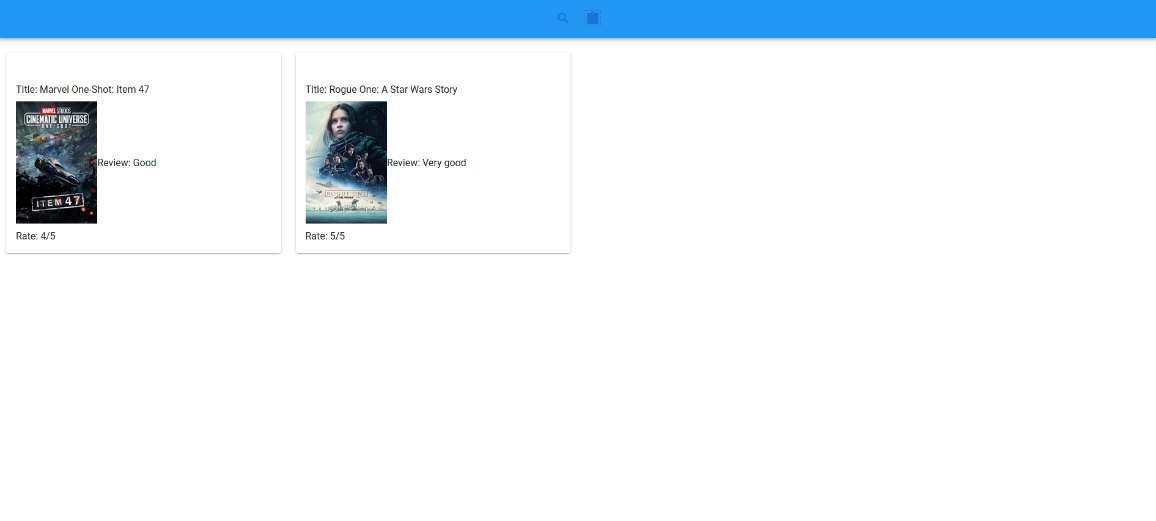
Le bouton vert « Opinion » affiche un popup pour critiquer un film avec un note et l’avis sur le film.



L’autre bouton, le bleu « movie detail » affiche un popup permettant de voir des informations a propos du film comme son titre, son affiche, son réalisateur et son résumé.



Une page nous permettant de voir toutes les critiques déjà réalisées par l’utilisateur connecté.

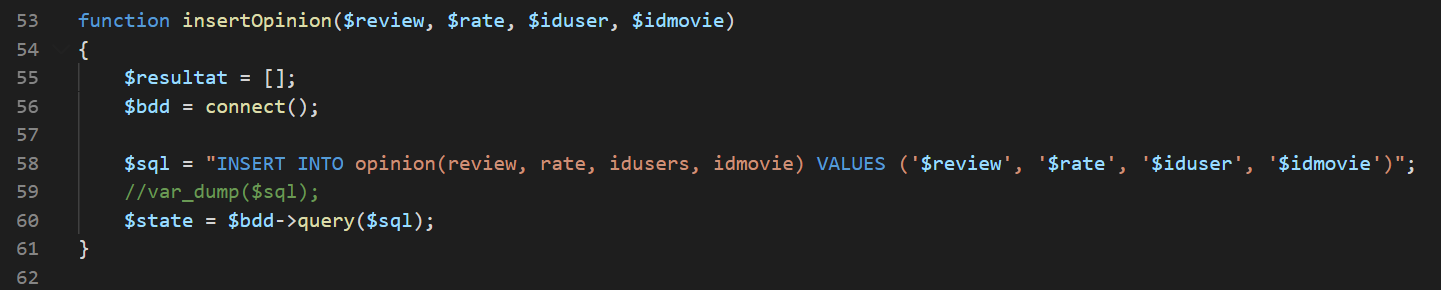


**Présentation du code de l’API REST:**

La méthode ci-dessous retourne un json dans lequel est présent l’id, le nom d’utilisateur et le mot de passe si les paramètres passés à la méthode correspondent à un compte utilisateur présent dans la table « user ».

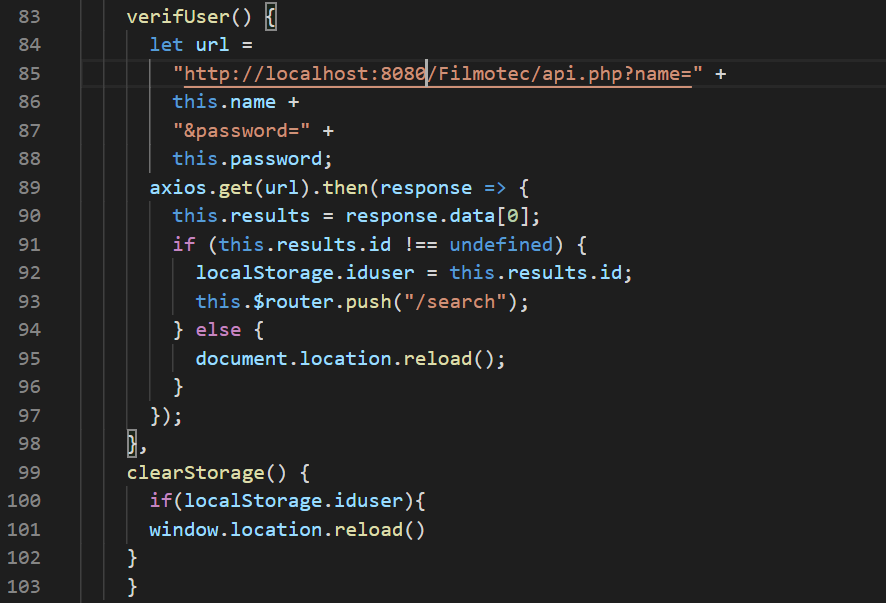


Cette autre méthode insert dans la table opinion une critique réalisée par un utilisateur avec comme paramètres l’avis, la note, l’iduser et l’idmovie.

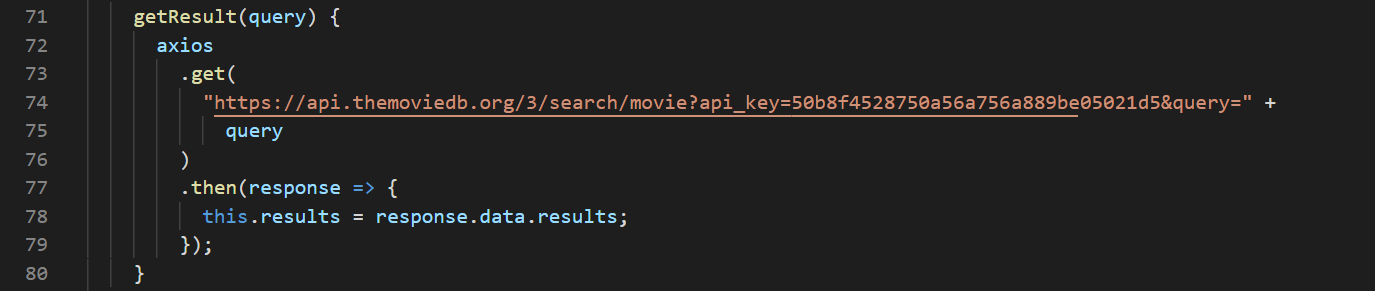


**Présentation du code vueJS :**

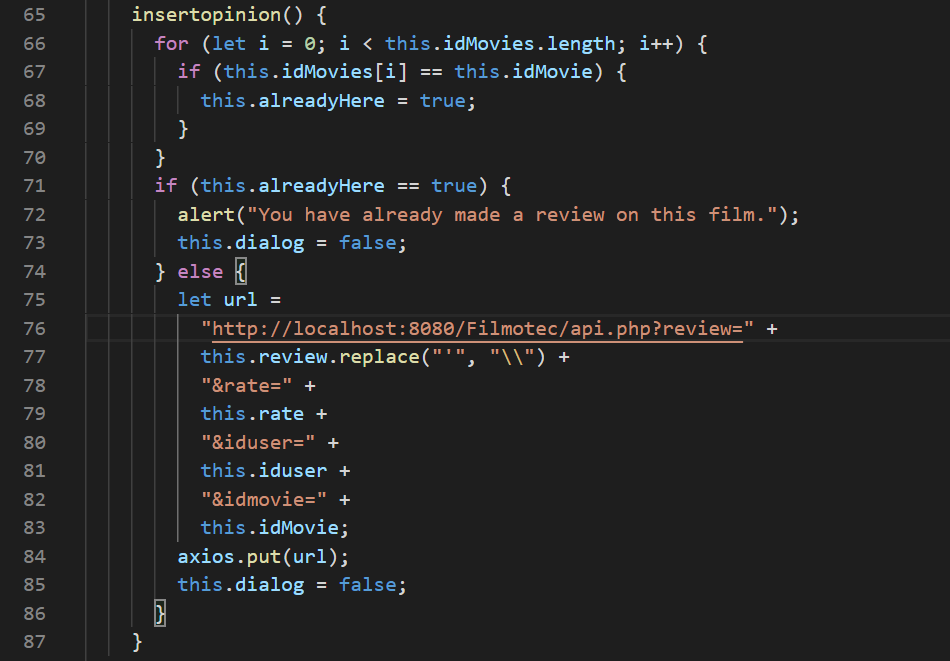
« verifUser » permet de vérifier si l’authentification à l’application est correcte grâce au nom d’utilisateur et mot de passe. La méthode interroge notre API REST et lui demande si le compte existe. S’il existe, la méthode nous autorise à accéder au cœur de l’application. Sinon on n’y accède pas.



« getResult » retourne les films dont le nom contient ce qui est entré dans le champ de recherche. Le paramètre « query » est alimenté par ce champ. La méthode appelle l’API TMDB avec comme paramètre « query » afin d’affiner la recherche. A la fin, la méthode alimente la variable « results » de la réponse de l’api TMDB.



Pour finir, la méthode « insertOpinion » insert la critique faite par l’utilisateur dans la table « opinion ». Tout d’abord, au début de la méthode on vérifie si l’utilisateur a oui ou non déjà critiqué ce film. Si oui un popup d’alerte l’averti. Sinon il peut le critiquer. Afin d’insérer la critique, la méthode appelle notre API REST avec comme paramètres l’avis, la note, l’iduser et l’idmovie. Enfin une fois l’insertion faite, le popup de la critique se ferme automatiquement.



**Conclusion :**

Le cahier des charges qui nous a été demandé est respecté. Notre application possède toutes les fonctionnalités demandées tels que la sélection de film à partir de l’API TMDB, la gestion des critiques dans un formulaire et la liste mensuelle de film. De plus, nous avons mis une page d’authentification fonctionnelle sur notre application ainsi que la possibilité de voir en détails les informations de chaque film.

Ce projet fut formateur mais aussi très instructif notamment dans l’apprentissage du langage vueJS que nous ne maitrisions que très peu au début du projet. Un bug non-résolu et persistant empêche l’utilisateur lors de la connexion de savoir qu’il n’a pas rentré le bon nom d’utilisateur ou mot de passe. Comme évolution côté API TMDB, il serait appréciable que celle-ci vérifie qu’elle possède bien toutes les affiches des films de sa base. En ce qui concerne notre application une évolution qui serait appréciée serait de rajouter la vidéo de bande d’annonce de chaque film dans le popup de détail du film.