Projet APNOTAN

Introduction/Présentation :

Ce document a pour but de décrire le déroulement du prototype de notre projet informatique portant sur un programme la mise en place application Web permettant l’évaluation de films.

Ce rapport est le résultat des recherches qui nous a permis de réaliser les prémices de ce projet.

Il contient l’ensemble des éléments de debut de realisation de notre projet.

Tout d’abord, nous présenterons le projet qui nous a été attribuer .

Nous expliquerons également l’utilité du choix d’un VPN.

Par La suite nous verrons le diagramme des cas d’utilisation

Nous explorons aussi le choix de notre support de réalisation ( vue Js )

Puis pour finir nous dévoilerons une première analyse de notre API REST

Nous espérons que vous prendrez autant de plaisir à lire ce rapport que nous en avons pris à le faire.

Note Sur le Choix d’un VPN :

Sans VPN, les données qui sont envoyé via Internet (surtout via les sites web qui ne respectent pas la norme HTTPS) sont visibles en clair et donc sans protection. Les pirates peuvent visualiser les données en transit et remonter leur piste pour identifier et accéder à l’adresse IP. Cela peut affecter directement l’entreprise si ces données incluent des informations confidentielles .

Indirectement, les utilisateurs du panel peuvent rendre vos serveurs et vos appareils vulnérables pour lancer des cyberattaques.

Choix Technique de Node/Vue :

Vue.js est facilement adaptable et peut être facilement intégré aux grands projets de développement de première ligne sans aucune difficulté. Sans plugins supplémentaires et bibliothèques externes, Vuejs aide à coder des applications robustes. De plus, il offre aux développeurs de bibliothèque de documents énorme, rendant ainsi le processus de développement d’applications Vue.js immensément plus facile.

Il est très flexible car il peut être utilisé dans n’importe quel environnement

Il offre aux développeurs une intégration facile avec des applications déjà existantes. Car il est basé sur le Framework JS.

En raison de son codage simple et facile, il conduit à une meilleure maintenabilité du code qui, en fin de compte, réduit les bugs et finalement nécessite moins d’heure de développement dans la correction de ces derniers , ce qui représente un gain de temps précieux au sein de projets ou le temps est l’une des principale contrainte

Analyse API REST:

Structure du JSON :

Get https://www.notreapplication.fr/liste

data {

Films {

“Nom Film” {

Title : “...”

Release Date : “...”

Auteur : “...”

Note Critique : “...”

Note Critique haute : “...”

Note Critique basse : “...”

etc…

}

“Nom film 2” {...}

}

Notre Api sera basée sera celle de <https://www.themoviedb.org/documentation/api>

Qui permet une utilisation par tous particulier comme professionnel

Et recense un nombre de film conséquent.

Diagramme UML des USE CASES:

FILMOTEC

<<Include>>

Afficher mensuellement une liste des films les mieux notés

Consulter la liste des films et leurs critiques

<<Include>>

S’authentifier

<<Include>>

Gérer les critiques

Membre du panel

<<Include>>

Récupérer les données des films depuis l’API

Récupérer les critiques

**Application APMAGWEB**

**Application APNOTPAN**

Utilisateurs

APMAGWEB

Sommaire :

I – Liste des films

*1.* *Récupération auprès de l’API*

II – Formulaire

*1.* *Formulaire de critique*

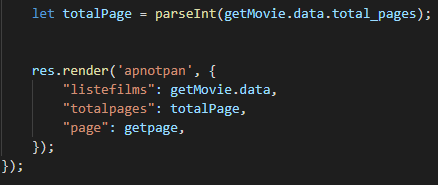
III –La base de données

1. *La récupération des données du formulaire*

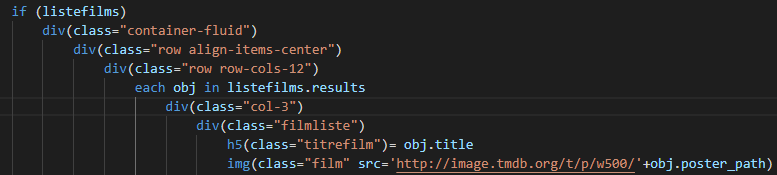
I – Listes des films

1. *Récupérations auprès de la base de données*

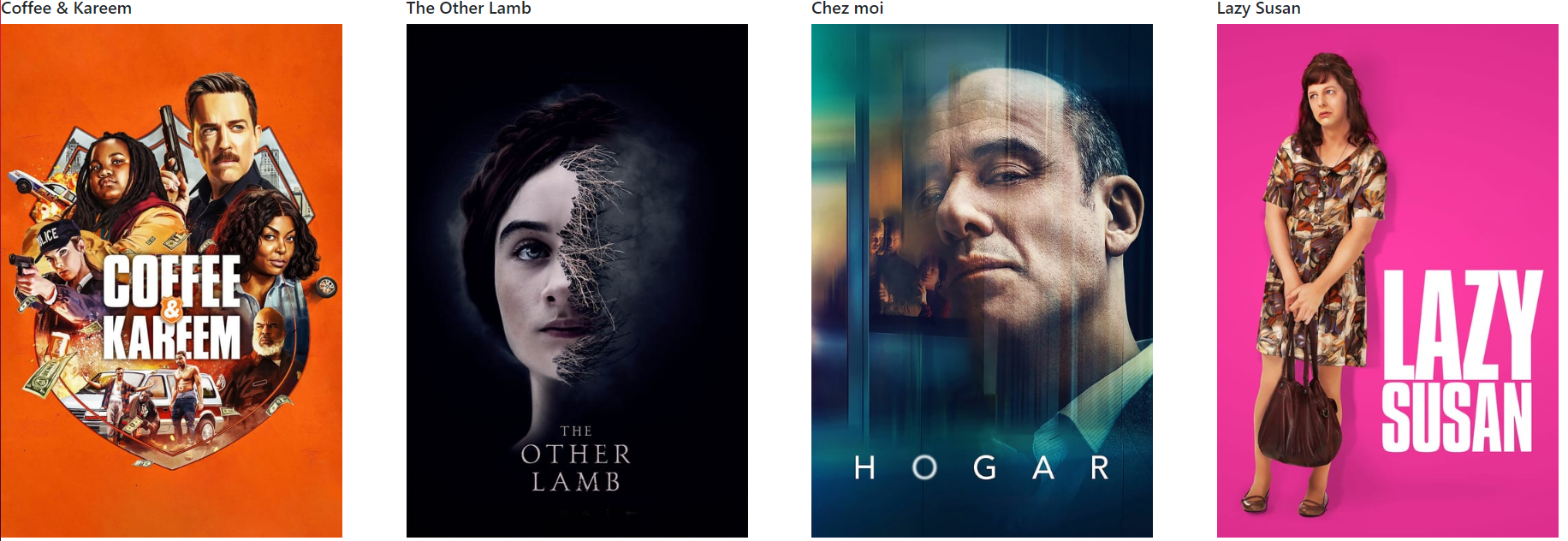
La Récupération des données de l’API TMDB se fait premièrement par la récupération d’une clé API voici la mienne « 65620f01c53f03f2a8384c231da22780 » une fois cette clé récupérer on récupère les données de l’api via un lien (Sachant que cette opération requiert le module Axios) :

Une fois les données stocker dans une variable on peut récupérer la liste des films qui est structuré par un tableau de page composé de 15 films par page. 

Simplement dans la vue il faudra donc faire une boucle qui parcourra chaque film et affichera son titre et son affiche



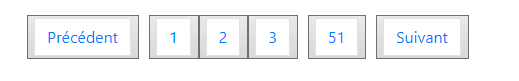
Ce qui donne en rendu :



Sachant que la page web se structure avec 4 films par ligne (Merci bootstrap)

Nous avons aussi modifier la taille des affiches pour que cela donne quelque chose d’assez uniforme

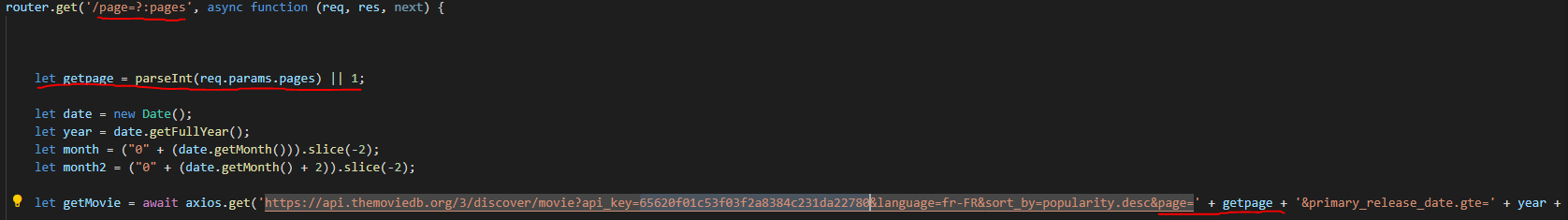
Nous avons énoncer le fait qu’ils y a différente page de 15 films hors le lien que nous stockons dans une variable ne renvoie que une seul page bien précise il a donc fallu créer une bar de navigation permettant de changer de page



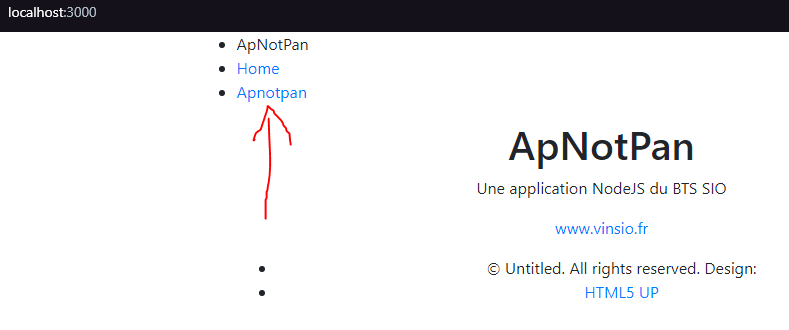
Voici le code dans la vue :



Une fois cet outil créer il fallait que celui-ci modifie directement le lien qui récupère la liste des films dans le controller :



Nous passons donc dans le lien de l’application la valeur de la page désirée qui sera par défaut 1 définit dans le layout.pug puisque a la base de l’appli nous nous trouvons ici il faut cliquer sur le bouton ApNotPan :

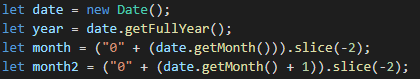


Et donc selon le bouton de la page sur laquelle on appuie la proposition de page change tout en laissant à disposition le retour a la première page et l’accès a la dernière page pour que l’utilisateur sache combien il y a de page de films au total :





Après cela nous faisons en sorte que l’appli se tienne à jour en termes de période automatiquement :



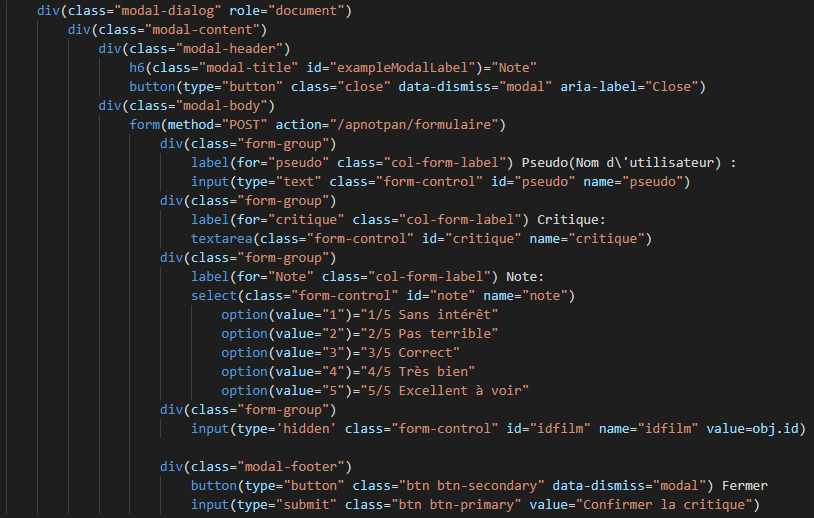
Nous récupérons ici le mois dernier et le actuel ce qui nous permet de récupérer tous les films d’actualité ce mois-ci en incluant dans le lien les valeurs récupérer :



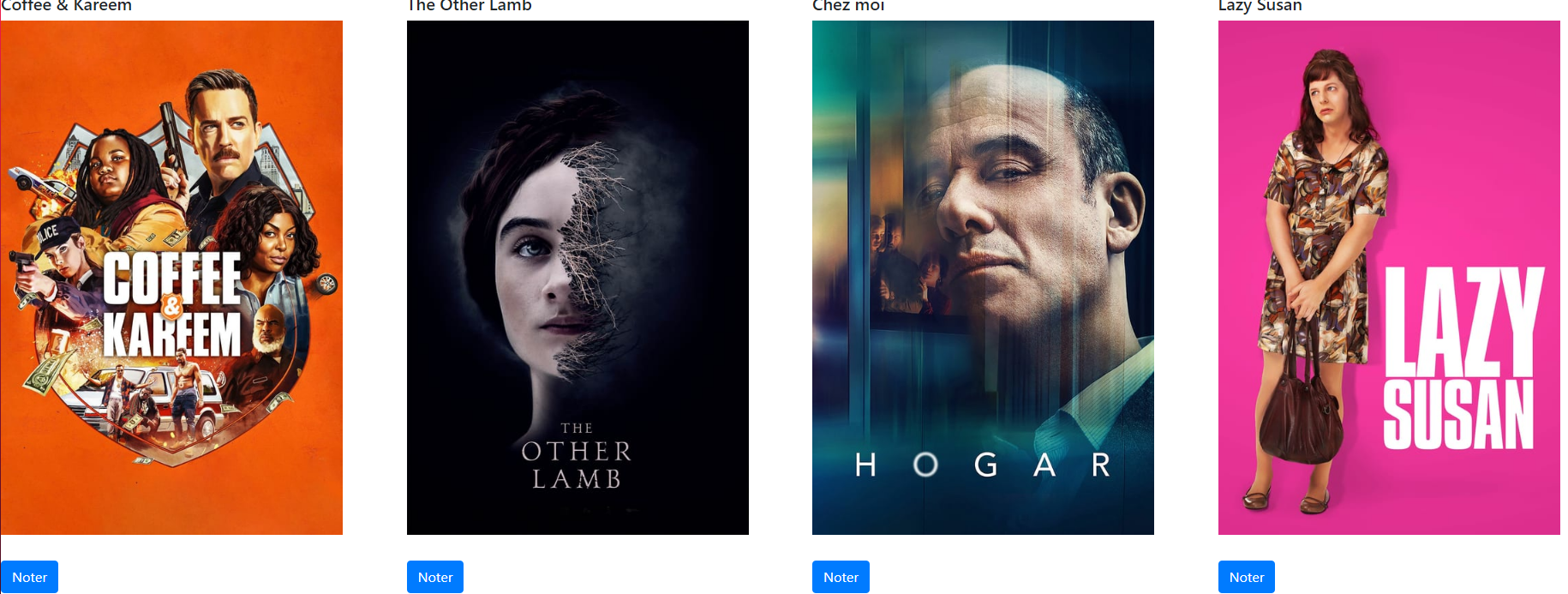
II – Formulaire

*1.* *Formulaire de critique*

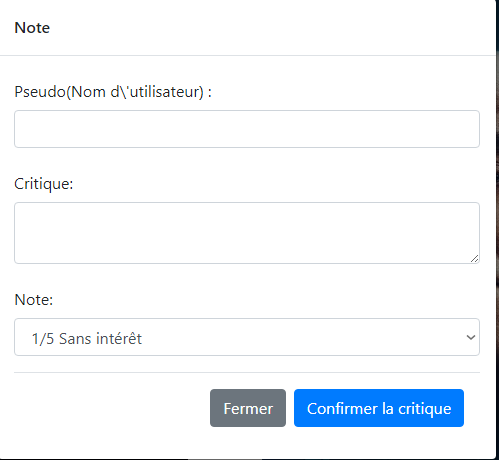
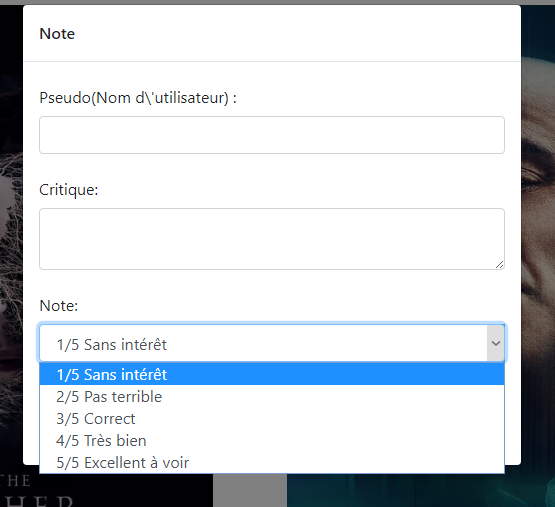
Pour le formulaire nous avons utiliser la fonction des modal de bootstrap :



Le formulaire étant disponible pour chaque film puisque ce code dans la vue se trouve dans la boucle qui affiche les films ce qui donne en rendu :



En cliquant sur le bouton noter cela nous affiche un pop-up permettant à l’utilisateur de s’identifier via un pseudo, de laisser sa critique et de noter le film avec une note allant de 1 à 5 :

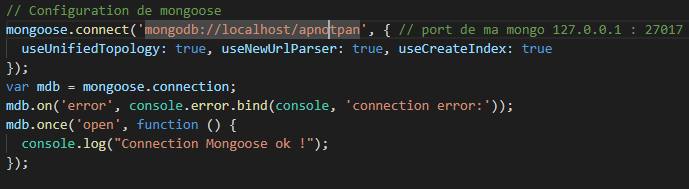


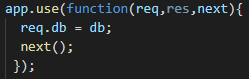
Voici les deux points de vue du formulaire puisque dans le premier les boutons fermer et confirmer sont cacher par le menu déroulant.

III – La base donnée

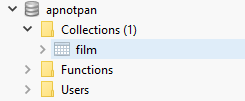
*1.* *La récupération des données du formulaire*

La technologie utilisée pour la base de données est mongodb donc sont utilisation avec nodejs requiert l’installation de monk et de mongoose :

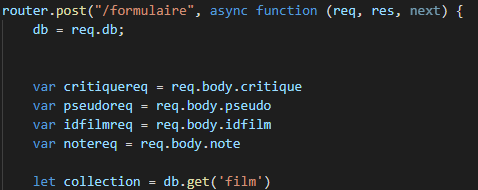


Une fois que tous ces modules sont ajoutés et que la base de données est créée avec sa collection appelé film :



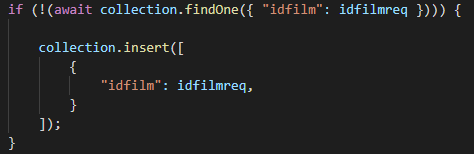
Nous pouvons donc commencer a utilisé la base de données.

Dans le screen précédemment soumis dans la section du formulaire celui montrant le code situé dans la vue nous pouvons identifier les value assigner aux différents champs celle-ci vont être utilisé pour envoyer les données dans la collection film de notre BDD :

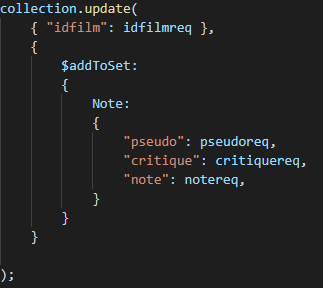


Nous voyons que les actions du code seront réalisées sur la route /formulaire car le bouton de confirmation de notre formulaire de critique renvoie sur cette route qui elle-même renverra l’utilisateur sur la page 1 de la liste des films (à noter qu’une amélioration future pourrait être faite pour que le formulaire renvoie l’utilisateur sur la page à laquelle il se situait)

Donc avec ces données que nous avons récupérées nous allons pouvoir faire des requêtes auprès de la BDD :



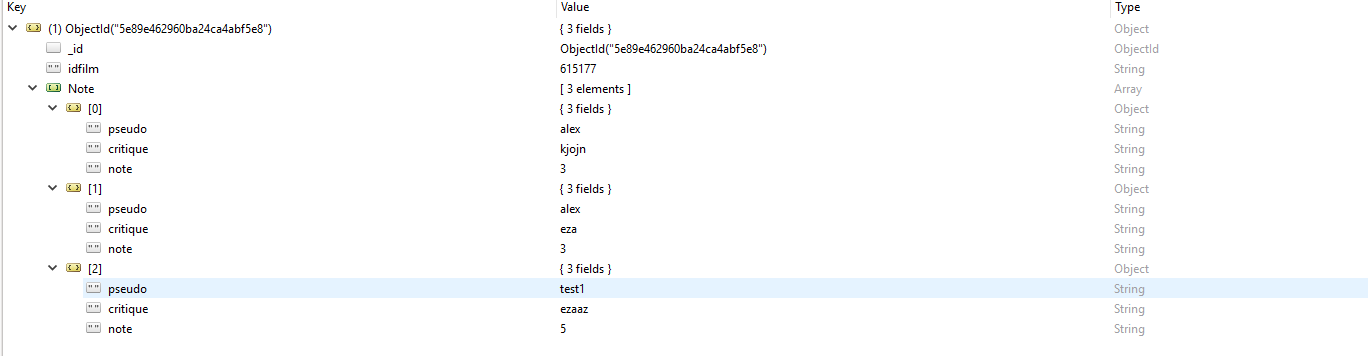
Premièrement nous vérifions si le film actuellement noté existe dans la BDD s’il n’existe pas nous créons un objet à son image identifiable grâce à l’id film.



A partir d’ici nous cherchons l’id film actuel dans la base de données sachant qu’il sera obligé d’exister puisque s’il n’existait pas il aurait été créer avec la requête précédente.

Ici nous créons un objet note dans l’objet identifier par l’id film dans cet objet note il y aura donc le pseudo que l’utilisateur aura choisis (en se disant que l’utilisateur utilisera toujours le même pseudo, une amélioration futur pourrait avoir lieu à ce niveau) ainsi que sa critique et sa note, la note qui sera identifier par un chiffre allant donc de 1 à 5 et non pas 1/5 – 2/5 – 3/5 …

La structure que cela donne dans la base de données est la suivante :



**Pour conclure**

Le projet n’est pas complètement terminé car l’utilisateur ne peux pas accéder à toute les critiques qu’il a postés sachant que la requête insérant la critique d’un utilisateur ne permet pas de voir si l’utilisateur en question avait déjà noté ce film et donc un utilisateur pourra noter plusieurs fois le même film ce qui est problématique encore un point à améliorer.

A part les fonctionnalités avec les critiques lié à la base de données le reste de l’appli est bien fonctionnel puisqu’elle liste bien tous les films d’actualité et se mettra automatiquement à jour selon l’année et le mois actuel en prenant une tranche de période qui va du 25 du moi dernier jusqu’au 31 du mois actuel.