1 SPA & JWT auths

1.1 SPA myMoovies avec JWT auths : lecture et ajout de films

L'application **myMoovies** doit permettre :

- D'authentifier un utilisateur ;
- L'ajout d'une ressource de type **films** via le formulaire d'ajout d'un film, seulement pour un utilisateur authentifié.
- L'affichage, sous forme de tableau, de toutes les ressources de type **films**.
- De gérer le routage des Pages et composants de manière moderne : affichage de la bonne page suite au clic sur la Navbar, affichage de l'URL dans le browser, sauvegarde de l'historique du browser, affichage du bon composant lors de l'utilisation de l'historique, gestion de l'affichage de la bonne page au chargement / refresh…
- D'afficher le nom de l'utilisateur authentifié ;
- De rediriger vers le composant de « Login » si l'utilisateur tente d'accéder à un composant nécessitant une authentification (ici le composant permettant d'ajouter un film) alors qu'il n'est pas authentifié.
- De permettre de déconnecter un utilisateur.
- D'avoir une Navbar qui affiche les bons liens en fonction que l'utilisateur est authentifié ou pas.

Vous allez mettre à jour le frontend de **myMoovies** réalisé à la fiche précédente (fiche 6). Vous utiliserez la RESTful API développée à la fiche 05 (API nécessitant une autorisation JWT) : cette API ne devrait normalement pas nécessiter de mise à jour.

Afin de réaliser cet exercice, voici les contraintes d'implémentation :

- Utilisez / exécutez la RESTful API **/js-exercises/fiche-05/mymoovies** incluant l'authentification et l'autorisation via JWT.
- Veuillez utiliser le proxy de votre frontend afin de contourner les problèmes associé à la gestion des CORS.
- Veuillez-vous assurer que vous avez un script pour chaque Page de votre frontend : **Homepage.js**, **Login.js**, **Register.js**…
- Veuillez utiliser le Router et la Navbar offerts dans la démonstration « Step 3 : Sauvegarde du token et du username dans le localStorage, affichage du username pour tout utilisateur authentifié & logout » js-demos/spa/spa-essentials/step3/ (ou dans le boilerplate d'exercices avec un routeur https://github.com/e-vinci/js-router-boilerplate) afin :

  - De charger le bonne page lors du chargement initial de la page ou d'un refresh.
  - De charger la bonne page lors d'un clic sur la Navbar.
  - De charger la bonne page lors de l'utilisation de l'historique du browser (Revenir en arrière ou Avancer).
- Veuillez sauvegarder côté client les informations de session pour tout utilisateur authentifié : le token et le username sont à sauver dans le localStorage.

- *Pour le frontend, nous vous recommandons de vous servir tant du frontend de la fiche précédente* **/js-exercises/fiche-06/mymoovies** *que des composants de la démo « Création d'une SPA pour une pizzeria : Step 3 : Sauvegarde du token et du username dans le localStorage, affichage du username pour tout utilisateur authentifié & logout » js-demos/spa/spa-essentials/step3/ (ou des composants du boilerplate d'exercices avec un routeur https://github.com/e-vinci/js-router-boilerplate)*

Veuillez-vous assurer de la bonne cohérence de votre UI, notamment :

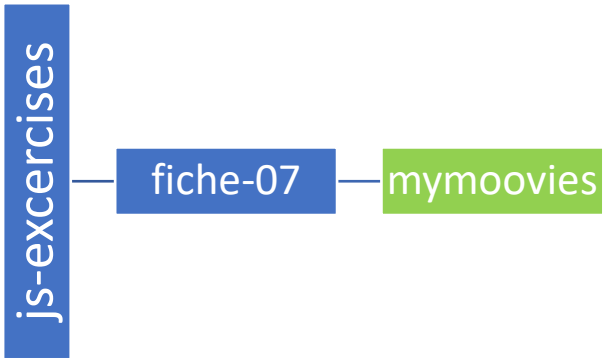
- La barre de navigation doit permettre d'accéder à toutes les pages.
- La Homepage devrait offrir la vitrine de vos films favoris (normalement réalisée lors de la fiche 3)
- Une page devrait permettre de lister tous les films qui ont été ajoutés via le formulaire ;
- Une page devrait permettre d'ajouter des films ; cela peut être la même page que celle listant les films. Néanmoins, si c'est le cas, veuillez vous assurer que le formulaire n'est pas visible pour les utilisateurs non authentifiés.
- …

Si vous avez une page spécifique pour l'ajout d'un film, vérifiez qu'un utilisateur non authentifié, qui tape l'URL du composant permettant l'ajout d'un Film, est redirigé vers la page de Login.

Tentez d'injecter du javascript à l'aide du formulaire d'ajout de films. Vérifiez que l'injection ne réussit pas car la RESTful API s'occupe d'échapper les caractères dangereux.

Une fois votre application terminée, veuillez faire un **commit** de votre code avec le message suivant : « myMoovies : step 8 : frontend integration with RESTful API and JWT auths : read and create films ».

Le code de votre application doit se retrouver dans ce dossier (en vert) de votre repository local et de votre web repository (**js-excercises**).



1.2 SPA myMoovies avec JWT auths : suppression et mise à jour de films

L'application **myMoovies** doit maintenant permettre ces nouvelles fonctionnalités :

- Effacer un film.
- Mettre à jour un film.

Pour ce faire, vous allez mettre à jour le frontend de **myMoovies** réalisé à l'exercice précédent (§1.1).

La page permettant de lister tous les films doit :

- soit être mise à jour afin d'autoriser la suppression et la mise à jour de films ;
- soit être mise à jour afin d'autoriser uniquement la suppression de films. Le clic sur un film en particulier pourrait rediriger l'utilisateur vers une autre page (ou une modal ou autre) détaillant ce film et permettant la mise à jour d'un film particulier.

Attention, « Effacer un film » et « Mettre à jour un film » ne doit être proposé que si l'utilisateur est authentifié.

Afin de réaliser cet exercice, voici les contraintes d'implémentation :

- Utilisez / exécutez la RESTful API **/js-exercises/fiche-05/mymoovies** incluant l'authentification et l'autorisation via JWT.

- *Pour les fonctions de mise à jour d'un film et de suppression d'un film, vous pouvez mettre à jour votre composant d'affichage des films. Voici un exemple de ce qui pourrait être présenté :*

Title	Link	Duration (min)	Budget (million)		
Harry Potter and the Philosopher's Stone	https://www.imdb.com/title/tt0241527/	152	125	Delete	Save
Avengers: Endgame	https://en.wikipedia.org/wiki/Avengers:_Endgame	181	181	Delete	Save
…					

- ***Option A pour les mises à jour d'un élément :*** *Si vous choisissez des mises à jour directement au sein du tableau présentant les films, vous pouvez utiliser la propriété HTML* ***contenteditable="true"*** *pour rendre un élément HTML editable. Ainsi vous pourriez utiliser cette propriété pour rendre les cellules du tableau éditables.*
- *Pour accéder aux cellules qui se trouvent dans une même ligne que vous mettez à jour, vous pouvez retrouver en JS :*
  - *Le parent d'un élément HTML via l'attribut parentElement ; par exemple, si vous avez un écouteur d'événements de clics sur un bouton « Save », ce bouton se trouvant au sein d'une td elle-même au sein d'une tr :* ***e.target.parentElement.parentElement*** *donne accès à la tr sur laquelle on a cliqué sur le bouton « Save »*
  - *Les enfants d'un élément HTML via l'attribut children ; par exemple,* ***tr.children[0]*** *donne accès à la première td au sein de tr*
- *Si vous aviez besoin de réaliser une action en cas de changement du contenu d'une cellule dont* ***contenteditable*** *est activé, vous pouvez gérer le type d'événement* ***input****.*
  Plus d'info : https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/input_event
- ***Option B pour les mises à jour d'un élément :*** *il est aussi possible de créer un nouveau composant Javascript (une page, une*

modal ou autre) qui reprendrait un formulaire dont les inputs contiendraient déjà les valeurs existantes des attributs d'un film.

- *Vous pouvez « cacher » de l'information, comme des ids, dans des tags HTML, via les attributs* ***data-***** : voir fin de [SPAs-Routage-Pages] (à partir du slide 14).*
- *Vous pouvez ajouter des écouteurs d'événements de manière dynamique à certains éléments, soit :*
    - ***Option 1*** *: lors de la construction de la table, si vous créez vous-même les éléments à attacher au DOM via* ***document.createElement()*** *et* ***appendChild()***
    - ***Option 2*** *: une fois la table rendue, après avoir allouée la propriété* ***.innerHtml*** *d'une div par une string représentant la table (ne jamais ajouter un écouteur d'événement avant d'allouer une valeur à* ***.innerHtml*** *apparenté).*
      N'hésitez pas à utiliser ***querySelectorAll()*** *pour obtenir un array de tous les éléments correspondant à une classe par exemple.*
- *Pour ne pas devoir ajouter trop d'écouteurs d'événements différents, vous pouvez utiliser un écouteur d'événement générique et l'associer à plusieurs éléments. Pour retrouver l'élément sur lequel l'événement a été soulevé, l' « event object » est automatiquement passé à la callback d'un écouteur d'événement : voir la slide « Event object » de [DOM-Events] (à partir du slide 29).*

Veuillez aussi tester ce qui se passerait si votre JWT est modifié côté client, au sein du Local Storage.

Une fois votre application terminée, veuillez faire un **commit** de votre code avec le message suivant : « myMoovies : step 9 : frontend integration with RESTful API and JWT auths : delete and update films ».

2 Exercice optionnel

2.1 SPA et appels

2.1.1 SPA myMoovies & auto-refresh d'infos

S'il vous reste du temps, il serait vraiment intéressant de faire en sorte que cycliquement, tous les 5 secondes par exemple, l'affichage de la liste de films se mette automatiquement à jour.

Attention que si le composant d'affichage des films est utilisé pour permettre de directement mettre à jour un film, celui-ci ne doit pas se réafficher si un utilisateur est en train de mettre à jour un film.

- *Besoin d'inspiration ? Nous vous recommandons la dernière slide de [DOM-Events] (**setInterval**).*

Une fois votre application terminée, veuillez faire un **commit** de votre code avec le message suivant : « myMoovies : step 10 : frontend integration with RESTful API and JWT auths : auto-refresh of film info ».

2.1.2 SPA myMoovies : gestion de la navigation lors de l'auto-refresh d'infos

Que se passe-t-il si un utilisateur passe rapidement de la page listant les informations d'un film vers la page d'ajout d'un film ?

Comme le render de tous les films et une action asynchrone, il est possible que celui-ci soit réalisé même quand l'utilisateur a quitté la page d'affichage des films. Cela peut être gênant, car si vous êtes sur le composant pour ajouter un film, vous ne souhaitez pas que votre formulaire d'ajout d'un film soit remplacé, de manière erronée, par la liste des films.

Nous vous proposons d'implémenter ces deux fonctions :

- Stopper les futurs appels de **fetch**, les appels automatiques vers vos APIs (**setInterval** et **clearInterval**) quand il y a un clic sur un lien ;
- Stopper les appels de **fetch** en cours vers vos APIs quand il y a un clic sur un lien. En effet, si votre API doit renvoyer beaucoup de données, notamment

quand il y a des images à communiquer, il est possible que cela prenne plusieurs secondes avant que l'API vous réponde.

Une fois votre application terminée, veuillez faire un **commit** de votre code avec le message suivant : « myMoovies : step 10+ : frontend integration with RESTful API and JWT auths : manage navigation and auto-refresh ».

- *Nous vous recommandons la dernière slide de [DOM-Events] (**setInterval** et **clearInterval**) pour stopper les futurs appels de fetch.*
- *Pour stopper un **fetch** en cours, vous pouvez utiliser un **AbortControlller**. Toute l'info ici : https://developer.mozilla.org/en-US/docs/Web/API/AbortController*