1 SPA monolithique avec JWT auth & bcrypt

Veuillez adapter la SPA développée pour l'exercice « 1 SPA monolithique & JSON » de la fiche 07 permettant d'enregistrer ,de manière persistante, des films et d'afficher les films enregistrés que pour les utilisateurs authentifiés.

Cette nouvelle application doit mettre en œuvre l'authentification et l'autorisation des ressources via des JWT, ainsi que le hachage des passwords.

De plus, cette nouvelle application doit permettre de mettre à jour les données associées à un film, ou d'effacer un film.

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

- Veuillez sécuriser toutes les fonctions de la RESTful API pour la collection de films : un JWT est nécessaire pour autoriser l'accès à ces ressources :

URL	Méthode	Fonction
films	GET	READ : Lire toutes les films de la collection
films/9	GET	READ : Lire le film dont l'id vaut 1
films	POST	CREATE : Créer (enregistrer) un film basée sur les données de la requête
films/9	DELETE	DELETE : Effacer le film dont l'id vaut 9
films/9	PUT	UPDATE : Remplacer l'entièreté de la ressource par les données de la requête

- Veuillez tester certaines fonctions de la RESTful API pour la collection de **films** à l'aide du REST Client dans VS Code (extension à installer au sein de VS Code) sans JWT, puis avec JWT.
  Veuillez ajouter vos requêtes au sein du fichier **film-requests.http** dans le répertoire **/restClient** du dossier associé à cet exercice.
- Via votre SPA, veuillez consommer ces ressources à l'aide la méthode **fetch()**, en utilisant **await** / **async** , et ajoutant un JWT dans le header des requêtes :
  - **GET films** pour la fonction d'affichage des films
  - **POST films** pour l'ajout d'un film via un formulaire
  - **DELETE films/:id** pour effacer un film via le tableau d'affichage des films

- **PUT films/:id** pour mettre à jour un film soit directement via le tableau d'affichage des films, soit via un formulaire reprenant les données actuelles associées au film

Une fois votre SPA fonctionnelle, veuillez tester ce qui se passerait si la durée de vie du JWT est configurée à 10 secondes.

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

- *Pour partir d'une application offrant l'enregistrement et l'authentification d'un utilisateur, la gestion de l'authentification et l'autorisation des ressources via JWT ainsi que le hachage des passwords, vous pouvez démarrer votre développement en copiant le contenu de la démo /demo/jwt dans votre dossier associé à cet exercice.*
- *Pour la gestion des films (ajout et liste de films), vous pouvez intégrer à votre projet le code de l'exercice « 1 SPA monolithique & JSON » de la fiche 07.*
- *Pour les fonctions de mise à jour d'un film et d'effacement d'un film, vous pouvez mettre à jour votre composant d'affichage des films. Voici un exemple de ce qui pourrait être présenté :*

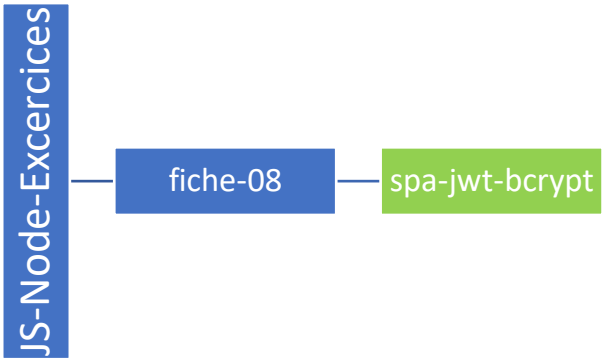
Title	Link	Duration (min)	Budget (million)		
Harry Potter and the Philosopher's Stone	https://www.imdb.com/title/tt0241527/	152	125	Delete	Save
Avengers: Endgame	https://en.wikipedia.org/wiki/Avengers:_Endgame	181	181	Delete	Save
…					

- *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.

- *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
- *Pour les modifications d'un film, il est aussi possible de créer un nouveau composant Javascript 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 dernier slide de [20-JS-Node-SPA-RoutingComposants.pdf].*
- *Vous pouvez ajouter des écouteurs d'événements de manière dynamique à certains éléments, soit :*
  - *lors de la construction de la table, si vous créez vous-même les éléments à attacher au DOM via* ***document.createElement()*** *et* ***appendChild()***
  - *une fois la table rendue, après avoir allouée la propriété* ***.innerHtml*** *d'une div par une string représentant la table. 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 [05-JS-Node-DOM-Events.pdf]*

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


JS-Node-Excercices
fiche-08
spa-jwt-bcrypt

2 Exercice optionnel : SPA frontend & backend indépendants

Si vous le souhaitez, vous pourriez adapter l'application que vous avez développée lors de l'exercice 1 de cette fiche afin d'avoir une application frontend indépendante de l'application backend.

Pour le frontend, il serait intéressant de mettre en œuvre Webpack. Afin d'ajouter une fonctionnalité sympa à votre SPA, vous pourriez essayer d'installer et utiliser un package de votre choix qui serait utilisé côté client.

De plus, vous pourriez ajouter la notion de thème via l'ajout d'une liste déroulante. Cette liste permettrait de sélectionner un thème de couleur pour votre application, en changeant, par exemple, les couleurs du header et du footer, ou d'autres éléments du layout.

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

- Utiliser le Local Storage pour gérer la notion de thème.
- Utiliser Webpack pour gérer votre frontend.
- Trouver un package qui permet de faire quelque chose vous tenant à cœur.

- *Pour partir d'une application offrant une bonne séparation du frontend et du backend, vous pouvez utiliser ou partir du contenu de la démo /demo/bundler dans votre dossier associé à cet exercice.*
- *Pour la gestion des films, vous pouvez intégrer à votre projet le code de l'exercice 1 de cette fiche.*

Le code de votre application devrait alors se trouver dans ce dossier (en vert) de votre repository local et de votre web repository (**JS-Node-Exercices**).