

420KBG – Laboratoire 03 – Authentification – client mono page

À partir des sources accessibles depuis *github* (voir plus bas), effectuer les tâches suivantes :

Côté API_SERVER :

https://github.com/Nicolas-Chourot/API_SERVER---V2

- Imposez une authentification pour les requêtes de verbe POST, PUT et DELETE s'adressant au service API_SERVER/api/bookmarks. Les requêtes GET dépourvues d'une clé d'accès (*bearer token*) pourront- être tout de même être servies.
- Ajoutez un champ "UserId" aux favoris indiquant l'utilisateur créateur du favori et modifier le service API_SERVER/api/bookmarks en conséquence. (i.e. GET retournera des objets du format suivant: {"Id": "...", "Name": "...", "Url": "...", "UserId": "...", "UserName": "..."}).
- Permettre la recherche ?username=... et permettre le tri ?sort=username.
- Ajoutez les chemins nécessaires dans la fonction API_SERVER/server.js/routeConfig({...}) pour atteindre les actions [PUT]accounts/change et [DELETE]accounts/remove.
- Complétez les actions du contrôleur AccountsController suivantes qui ne pourront être servies qu'avec une clé d'accès:
 - o login(loginInfo) inclure dans la réponse non seulement la clé d'accès mais aussi le Id de l'utilisateur et son nom. (le client devra conserver ces informations localement).
 - o change(user) pour permettre le changement de profil incluant le mot de passe.
 - o remove(id) pour permettre le retrait d'un compte utilisateur. Note : un retrait d'utilisateur doit effacer tous ses favoris.

Côté BookmarksManager (le client Web) :

<https://github.com/Nicolas-Chourot/BookmarksManager---v2>

- Ajoutez au client BookmarksManager :
 - o Un dialogue d'enregistrement d'utilisateur (nom, courriel et mot de passe).
 - o Un dialogue d'authentification avec un courriel et un mot de passe.
 - o Un dialogue de modification de profil (nom, courriel et mot de passe).
 - o Un dialogue de confirmation de déconnexion.
 - o Ajouter à l'interface le nécessaire pour accéder aux dialogues précédents.
 - o Ajouter à l'interface le nom de l'utilisateur qui est connecté.
 - o Une redirection vers le dialogue de connexion pour toutes tentatives d'écriture (ajout, modification et retrait de favori) lorsque la clé d'accès de l'utilisateur est expirée.
 - o Ajoutez une colonne « user » qui indique le créateur du favori s'il est différent de celui qui est connecté, sinon, rien inscrire.
 - o Ajoutez un champ de recherche par nom d'utilisateur
 - o Rendez inaccessible les accès à l'ajout, modification et retrait pour les favoris qui n'ont pas été créés par l'utilisateur connecté.

Hébergez votre version modifiée d'API_SERVEUR dans un répertoire glitch ainsi que votre version du client Web BookmarksManager dans un autre répertoire glitch. Des points bonus seront accordés si vous imposez correctement l'origine des requêtes spécifiquement au client.

Remise : 9 novembre avant la rencontre de 16h15

Livrables :

- Un fichier zip de Windows (pas 7zip, rar, etc.) portant les noms de famille des participants. Par exemple Tremblay-Lavoie.zip. Lorsque l'on « *dézip* » on doit découvrir un répertoire portant les noms des participant qui inclus deux répertoires, API_SERVER (incluant tous les sources du serveur) et BookmarksManager (incluant tous les sources du client)
- Les liens vers les répertoires glitch du serveur et du client.

Annexe – technique :

Dialogues

Consultez le répertoire suivant pour voir comment créer des dialogues en JavaScript et jQuery :

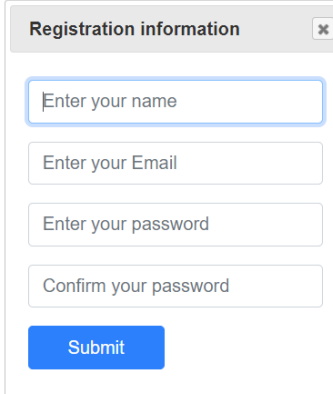
<https://github.com/Nicolas-Chourot/dialogDemo>

Enregistrement local

Stockage de clé d'accès avec les fonctions intégrées des fureteurs :

`localStorage.setItem(clé, valeur)`

`localStorage.getItem(clé)`

A registration form titled "Registration information" with a close button (X). It contains four text input fields: "Enter your name", "Enter your Email", "Enter your password", and "Confirm your password". A blue "Submit" button is at the bottom.

```
let apiBaseUrl = "http://localhost:5000";

function eraseAccessToken() {
    localStorage.setItem('access_token', null);
}

function retrieveAccessToken() {
    return localStorage.getItem('access_token');
}

function getBearerAuthorizationToken() {
    return { 'Authorization': 'Bearer ' + retrieveAccessToken() };
}

function storeLoggedUsername(username) {
    localStorage.setItem('username', username);
}

function retrieveLoggedUsername() {
    return localStorage.getItem('username');
}
```