Université de Montréal

COURS IFT1142

TRAVAIL PRATIQUE #2

CONSIGNES

1. **Il faut remettre la version électronique. On n'accepte pas des fichiers envoyés par courriel.**
2. **La date de remise est sur Studium.**
3. **Tout travail remis en retard aura une pénalité de 10% par jour de retard.**
4. **Les travaux remis avec 5 jours et plus de retard seront refusés.**

GESTION D'UN CATALOGUE DE LIVRES VIA WEB AVEC UN FICHIER LIVRES.JSON

1. PRÉSENTATION DU TRAVAIL PRATIQUE

Ce travail pratique 2 est une reprise du travail pratique 1 mais dans un contexte Web. Nous allons concevoir une **WebApp** (application Web) en mode **SPA** (Single Page Application). Nous allons faire des requêtes asynchrones via **fetch** à notre serveur Node. Toutes les opérations se feront dans un fichier **livres.json** côté serveur et les réponses sont retournées au client en format JSON.

Nous allons utiliser **Bootstrap** comme « framework » pour la réalisation de nos éléments de HTML et CSS en se garantissant que notre page Web reste toujours responsive.

Titre : Gestion de livres (CRUD et requêtes de sélection)
Langage : JavaScript ES6+
Données : Fichier **livres.json**
Technologies :

- **Page Web responsive**
- **JavaScript ES6+, serveur Node**
- **Fichier livres.json** pour stocker les livres.

Le travail consiste à concevoir une application qui permet de **créer**, **lire**, **modifier** et **supprimer** des livres, ainsi que d'effectuer **quelques requêtes de sélection** (exemples plus bas).

2. STRUCTURE DES DONNÉES DANS LIVRES.JSON

Chaque livre est représenté par un objet JSON de la forme suivante.

```
[
 {
  "id": 400,
  "titre": "Une aventure d'Astérix le gaulois. Le devin",
  "idAuteur": 11,
  "annee": 1972,
  "pages": 48,
  "categorie": "bandes dessinées",
  "pochette" : "images/pochettes/nom_image.extension"
 },
```

Pour la pochette vous pouvez trouver les images en tapant le nom du livre sur Google.

Explication des attributs d'un objet livre :

- **id** (entier) : identifiant unique du livre.
- **titre** (texte) : titre du livre.
- **idAuteur** (entier) : identifiant de l'auteur (ou tout autre code d'auteur).
- **annee** (entier) : année de publication.
- **pages** (entier) : nombre de pages du livre.
- **categ** (texte) : la catégorie du livre (ex. "bandes dessinées", "roman", "manga", etc.).
- **pochette** (texte) chemin où se trouve l'image dans le projet

Tous les ajouts, modifications ou suppressions doivent être enregistrés dans le fichier **livres.json** qu'est du côté serveur et cela à chaque chagement.

3. CONTENU ATTENDU (FONCTIONNALITÉS)

1. **Barre de navigation avec les options**
    - Logo à gauche et après les options :

Ajouter	Lister	Rechercher (select)	Trier
		Catégorie	
		Année	
		Auteur	

2. **CRUD (sur le fichier livres.json) :**
    - **C** (Create) : Ajouter un livre en demandant les informations via un formulaire modal de Bootstrap.
    - **R** (Read) : Lister tous les livres, afficher leurs informations via des « cards » de Bootstrap. Chaque card doit avoir en fin de celle, deux icônes un pour modifier (crayon) et un pour supprimer (poubelle). Utilisez les icônes de Bootstrap.
    - **U** (Update) : Lorsqu'on clique sur l'icône crayon, placer les données du livre en question dans un formulaire et y apporter les modifications.
    - **D** (Delete) : Lorsqu'on clique sur l'icône poubelle, affichez un Toast de Bootstrap comme message de confirmation (Oui ou Annuler).
3. **Requêtes de sélection :**
   Vous devez implémenter les trois requêtes (via les options du menu) pour filtrer ou sélectionner des livres, par exemple :
    - Sélection par **catégorie** (afficher uniquement les livres d'une catégorie donnée).
    - Sélection par **année** (afficher uniquement les livres publiés après une certaine année).
    - Sélection par **auteur** (selon l'idAuteur, etc.).

L'affichage du résultat de chaque requête se fera selon le format suivant :

De « cards » de Bootstrap, 4 par ligne. Des exemples vous seront donnés en classe.

4. **Ergonomie :**
    - Gestion d'un menu via la console (terminal).
    - L'interface doit être **intuitive** et **cohérente**.
5. **Barème d'évaluation (sur 100%)**

Critères	Points
Ergonomie	10%
– Utilisation efficace de la barre de navigation	
– Navigation claire et cohérente	
– Présentation des résultats et demande de saisie de données	
Fonctionnalités CRUD & Requêtes (7x10%)	70%
– Ajout, Modification, Suppression	
– Liste correct des livres via des cards et ses icônes	
– Trois requêtes de sélection	
Structure du code	20%
– Code commenté	
– Utilisation du fichier **livres.jon**	
– Respect de la structure de projet demandée.	

Critères	Points
– Clarté et organisation (fonctions, modules, utilisation des méthodes de la classe Array de JavaScript, routes côté serveur, opérations de CRUD sur le fichier livres.json, etc.)	
Total	**100%**

5. PÉNALITÉS SPÉCIFIQUES

1. **-10%** par consigne non respectée (ex. pas de menu, format d'affichage non respecté, etc.).
2. Toute option de traitement qui ne fonctionne pas **= 50% maximum** des points attribués à la question, l'autre 50% étant accordé pour la partie du code qui a été implémentée.

Exemple : Si la requête de sélection par catégorie est un des points notés à 10, mais l'option du menu ne fonctionne pas, vous obtiendrez au maximum 5 points, et seulement si le code interne pour filtrer les livres est présent et bien fait.

6. LIVRABLES ATTENDUS

- Le dossier du projet nommé TP2 selon la structure de projet donnée en classe.
- Le projet doit être remis dans Studium dans le lien REMISE TP2

7. CONSEILS / RECOMMANDATIONS

- **Commenter** votre code : chaque fonction doit être documentée, ainsi que les blocs les plus importants.
- Respecter le format du projet.
- Testez fréquemment vos opérations :
  - Ajouter un livre,
  - Vérifier qu'il est bien écrit dans livres.js,
  - Modifier ce même livre,
  - Supprimer ensuite, etc.
- Restez **cohérent** sur les noms de variables et gardez ce même nom partout dans votre projet (pensez au dictionnaire de données), la structure, la navigation, etc.

EN RÉSUMÉ :

Vous devrez produire une **application Web avec Node** permettant de gérer une liste de livres (fichier **livres.json**), avec un **CRUD complet** (Create, Read, Update, Delete) et **trois requêtes de sélection** (filtrage). L'**interface** d'accueil sera réalisée via une barre de navigation qui apparait dans votre page index.html. Vous serez évalué sur l'**ergonomie** (10%), la **qualité et l'exhaustivité** des fonctionnalités (70%), et la **structure et clarté du code** (20%), tout en respectant les consignes pénalisantes mentionnées.

Bonne réalisation et bon courage ! 😊