LOG2440 – Méthod. de dévelop. et conc. d'applic. Web

Travail pratique 5

Chargés de laboratoire:

Thierry Beaulieu

Rachad Chazbek

Charles De Lafontaine

Jamesley Joseph

Antoine Poellhuber

Automne 2022

Département de génie informatique et génie logiciel

1 Objectifs

Le but de ce travail pratique est de vous familiariser avec le développement *full stack* : système complet qui comporte un site web, un serveur dynamique et base de données. Vous utiliserez *React* pour le site web et la base de données MongoDB à partir d'un serveur NodeJS/Express. Cet ensemble est appelé le *MERN stack (Mongo/Express/React/Node)*.

Plus spécifiquement, vous aurez l'occasion à modifier la manière de gérer la persistance des données de votre système en remplaçant les fichiers JSON du serveur par l'utilisation d'une base de données MongoDB. De plus, vous serez initiés au développement de sites web dynamiques à l'aide de librairies spécialisées et la programmation déclarative grâce à React.

2 Introduction

Au cours de cette session, vous avez mis en place un site web à l'aide des technologies de base du web : HTML, CSS et JS, puis vous y avez ajouté un serveur dynamique en utilisant NodeJS et la librairie Express. Vous aviez donc un système fonctionnel dont les informations pouvaient être partagées entre plusieurs utilisateurs.

Les technologies utilisées lors de ces travaux pratiques sont tout à fait valides et encore présentes de nos jours. Cependant, il existe maintenant plusieurs outils permettant une meilleure gestion des différents éléments présents sur une page web et facilitant la mise en place d'une architecture logicielle fiable. C'est le cas de la librairie React basée sur l'implémentation de composantes réutilisables ainsi que sur l'utilisation d'un DOM virtuel. Pour ce dernier travail pratique, nous avons donc réécrit le code du côté client de votre site web à l'aide de cette librairie et vous serez amenés à le compléter.

Comme au TP4, il vous faudra démarrer à la fois un serveur statique qui sert votre application React et votre serveur dynamique NodeJS. Le serveur *Webpack* remplacera *lite-server* utilisé à date.

Au TP4, vous avez assuré la persistance des données grâce à des fichiers JSON. Comparativement à cette méthode, l'utilisation d'une base de données comporte plusieurs avantages.. Pour assurer la persistance de vos données, vous allez donc devoir faire usage de la base de données orientée documents MongoDB. Le serveur dynamique du dernier travail pratique a été modifié pour assurer la liaison entre votre site web et votre base de données.

Après avoir terminé l'implémentation du serveur et la connexion à la base de données, le serveur du TP5 devra être capable de fonctionner avec le site web du TP4.

3 Configuration du projet

3.1 Syntaxe JSX et serveur de développement

La particularité de la librairie React est l'utilisation de la syntaxe JSX utilisée pour aider avec l'approche déclarative de développement et qui mélange l'utilisation de JS et HTML dans le même fichier. Contrairement à JS et HTML, JSX n'est pas supporté par les navigateurs et doit être transpilé en JS/HTML natifs avant d'être interprété par un navigateur.

Pour vous aider dans votre développement, vous utiliserez l'utilitaire *react-scripts* qui est un ensemble d'outils qui simplifient le développement de projets React. Par exemple : `Babel` pour la transpilation et `Webpack` pour la minification et le déploiement. L'outil surveille également pour des changements dans le code source du projet et le relance automatiquement (*hot-reload*). Dans votre cas, ceci permettra d'automatiquement relancer le serveur statique lorsque vous modifiez son code source sans que vous ayez à le faire à la main.

Pour lancer le serveur statique, vous pouvez utiliser la commande npm **start** dans votre terminal. Cette commande va transpiler le code JSX avant de lancer le serveur statique. Par défaut, ce serveur est accessible à l'adresse : "http ://localhost :5010".

Votre serveur dynamique NodeJS est disponible à travers la même commande lancée dans son répertoire. Le serveur dynamique sera accessible à l'adresse : "http ://localhost :5020".

3.2 Tests automatisés

Plusieurs tests vous sont déjà fournis dans le répertoire `server/tests` et vous permettront de valider l'interaction entre le serveur et MongoDB. Les tests unitaires utilisent la librairie *Jest*. La librairie *MongoMemoryServer* est utilisée pour remplacer l'instance de MongoDB par une instance contrôlée pour les tests. **Note** : Il est très important de se connecter à une BD locale et non à votre instance MongoDB en mode production pendant vos tests.

Pour lancer les tests, allez à la racine du répertoire `server` avec un terminal et lancez la commande **npm test**. Les tests lanceront votre serveur et feront plusieurs requêtes vers l'instance de MongoMemoryServer avant de le fermer à la fin de la suite. Une commande pour la couverture de code (**npm run coverage**) est également à votre disposition.

Il est fortement recommandé de regarder les tests fournis et leur description avant d'écrire votre code. Vous pouvez ajouter vos propres tests si vous considérez que c'est pertinent.

4 Travail à réaliser

Contrairement aux TPs précédents, il n'y a aucune nouvelle fonctionnalité à ajouter à votre site web. Vous allez vous concentrer sur l'intégration d'une base de données MongoDB comme remplacement aux fichiers JSON utilisés au TP4 ainsi que la refonte de votre application client (site web) à l'aide de la librairie React. À la fin du travail, votre site web doit avoir le même visuel et le même comportement que votre travail pratique 4.

La première partie de votre travail est de déplacer vos données sur une instance MongoDB. Consultez le fichier README de l'exemple du cours afin de créer votre instance MongoDB sur le service *Cloud Atlas* qui hébergera votre base de données en ligne. **Rappel :** MongoDB peut contenir plusieurs collections. Pour ce TP, on vous demande de placer les différentes données dans 2 collections séparées nommées *playlists* et *songs*.

Vous aurez également à transformer votre site web en un projet utilisant la librairie React. Une partie du travail est déjà faite pour vous et vous est fournie comme code de départ. Vous devez compléter le reste pour retrouver l'ensemble des fonctionnalités de votre projet.

4.1 Communication avec la base de données

Afin de mieux s'accorder avec une architecture orientée services (*SOA*), des classes du serveur du TP4 ont été renommées, notamment `PlaylistService` et `SongService`. La classe `DatabaseService` a été rajoutée et la gestion des données aura désormais lieu dans MongoDB.

Vous devez implémenter les fonctionnalités présentes dans ces 3 fichiers et n'avez pas besoin de modifier les contrôleurs ni la configuration du serveur. Chaque fonction à implémenter contient le mot clé **TODO** dans son en-tête. Les fonctions à compléter retournent des valeurs vides afin de permettre l'exécution du code, mais vous devez modifier ces valeurs de retour.

Consultez la documentation de MongoDB ainsi que celle du NodeJS Driver pour les différentes méthodes et leurs valeurs d'entrée/sortie.

4.1.1 DatabaseService

Le fichier 'env.js' contient les constantes avec les informations de connexion à votre instance MongoDB. Vous devez les remplacer avec les bonnes valeurs de votre propre instance.

Vous devez implémenter la fonction `populateDb` qui remplit une collection avec des données seulement si la collection est vide. Cette fonction est exécutée lors du lancement du serveur et remplit la BD avec les valeurs des 2 fichiers JSON du répertoire `data` du projet. Si les collections de la BD contiennent déjà des valeurs, aucune modification n'a lieu.

4.1.2 PlaylistService

Ce service s'occupe des playlists et leurs manipulations sur la base de données.

Vous devez implémenter les fonctions suivantes : `getAllPlaylists`, `getPlaylistById`, `addPlaylist` et `updatePlaylist`. Consultez les en-têtes des fonctions ainsi que les tests présents dans `playlist.service.test.js` pour bien comprendre le fonctionnement des méthodes. La méthode `deletePlaylist` vous est fournie à titre d'exemple. L'accesseur utilitaire `collection` permet d'avoir accès à la bonne collection de la base de données.

4.1.3 SongService

Ce service s'occupe des chansons et leurs manipulations sur la base de données.

Vous devez implémenter les fonctions suivantes : `getAllSongs`, `getSongById`, `updateSongLike` et `getRecipesByIngredient`. Consultez les en-têtes des fonctions ainsi que les tests présents dans `songs.service.test.js` pour bien comprendre le fonctionnement des méthodes. L'accesseur utilitaire `collection` permet d'avoir accès à la bonne collection de la base de données.

4.1.4 SearchBarService

Le code de la fonction `search` vous est déjà fourni. Vous devez compléter les fonctions `search` dans `SongService` et `PlaylistService` qui vont effectuer la recherche par mot clé.

Le code initial fourni dans ces 2 méthodes effectue une recherche seulement sur l'attribut `name` et n'est pas sensible à la case (paramètre : `$options: "i"`). Vous devez modifier la recherche pour tenir compte des autres champs possibles et tenir compte du paramètre `exact` (recherche sensible à la case). **Astuce** : l'opérateur logique $or de MongoDB vous sera utile.

4.2 Implémentation du site web avec la librairie React

Vous devez compléter le code fourni pour compléter votre site web en utilisant la librairie React. Contrairement aux TPs précédants où chaque page est implémentée dans un fichier HTML séparé, ce travail utilise les composantes React pour construire les différents éléments de l'interface graphique. Vous utiliserez seulement des composantes fonctionnelles de React.

Vous allez utiliser le patron *SAM* et le concept de `Reducer` pour gérer les *actions* envoyées par les composantes et modifier l'état de la playlist en cours. Chaque action contient obligatoirement un type et, optionnellement, un contenu (*payload*). Par exemple : `{type: ACTIONS.PLAY, payload : { index: 1}}`. Le fichier `reducer.js` contient le `Reducer` utilisé. Lisez bien le code fourni pour comprendre la bonne formulation des actions à envoyer.

4.2.1 Composantes de pages

Le répertoire `pages` contient l'ensemble des composantes React qui définissent les différentes "pages" de votre site web et sont similaires aux fichiers HTML des TPs précédants. En réalité, vous aurez une application monopage (*Single Page Application*) avec un seul document HTML dont le contenu est chargé dynamiquement. La configuration des routes à l'aide de la librairie `react-router-dom` vous est fournie dans le fichier `App.jsx`.

La composante `About` vous est fournie et vous n'avez pas à la modifier. Les autres composantes contiennent le mot clé **TODO** aux endroits à compléter.

Index : vous devez compléter la recherche par mot clé dans la fonction `handleSearch`. Notez que cette fonction est utilisée par la composante `SearchBar` et doit lui être passée. Vous devez compléter la récupération des chansons et leur affichage à l'aide de composantes `Song`. La gestion des playlists à l'aide de composantes `Playlist` vous est donnée à titre d'exemple.

Playlist : vous devez récupérer la référence vers la classe `HTTPManager` partagée à travers le `PlaylistContext` pour faire des appels au serveur. Notez que la fonction `fetchData` utilise `dispatch` pour charger les chansons dans le `Reducer`. Vous devez compléter le HTML de la composante pour le chargement des chansons. Notez que contrairement à `Index`, les chansons ici ont un affichage et un comportement différent contrôlé par la propriété `index` comme sera expliqué plus bas. Vous devez ajuster le code de génération de composantes en conséquence.

CreatePlaylist : similaire au TP4, le formulaire permet de créer une nouvelle playlist ou en modifier une existante si la "page" est accédée avec un `id` dans l'URL. La variable `data` contient l'information de la playlist à envoyer au serveur. Vous devez compléter les fonctions `handleNameChange` et `handleDescriptionChange` qui gèrent la modification du nom et la description de la playlist. Vous devez compléter le code dans les fonctions `handleSubmit` et `deletePlaylist` qui effectuent des requêtes à votre serveur pour supprimer ou ajouter/modifier une playlist dans le cas de `handleSubmit`. Vous devez compléter le HTML de la composante aux endroits indiqués, notamment la construction dynamique des éléments `<option>` pour le choix de chansons, la liaison des entrées de nom et description ainsi que la modification du bouton de soumission en fonction de type de formulaire utilisé.

4.2.2 Autres composantes React

Le répertoire `components` contient l'ensemble des autres composantes React qui définissent les différents éléments utilisés dans les composantes de pages du site web.

Les composantes `SearchBar` et `Footer` vous sont fournies pour vous aider avec votre travail et vous n'avez pas à les modifier sauf pour ajouter vos noms dans `Footer`.

Les autres composantes contiennent des éléments à compléter. Le mot clé **TODO** est présent aux endroits à compléter. Vous devez compléter des fonctions ou compléter le HTML retourné par la composante en utilisant la syntaxe JSX.

NavBar : vous devez ajouter les bons liens vers les pages `/index` et `/create_playlist`. Le lien vers la page `/about` vous est déjà fourni.

Song : vous devez compléter la fonction `toggleLike` qui effectue une demande de changement du statut aimé d'une chanson à votre serveur et modifie l'affichage. Cette fonction doit être disponible sur le bouton de la composante seulement lorsque la propriété `index` n'est pas présente (affichage dans la librairie). Vous devez compléter la fonction `playSong` qui envoie une action `PLAY` au `reducer` and le bon index. Cette fonction doit être disponible seulement lorsque la propriété `index` existe (affichage dans une playlist). Vous devez compléter le code HTML pour afficher les informations de la chanson, notamment son nom, genre et artiste.

Playlist : vous devez compléter le code HTML pour les informations de la playlist choisie, notamment son image, son nom et son description. Notez que même s'ils partagent le même nom, cette composante est différence de la composante `Playlist` dans le répertoire `pages`.

Player : vous devez compléter les différentes fonctions de contrôle de musique. Notez que la musique en cours est contrôlée par un `reducer` et la composante ne fait qu'envoyer les actions appropriées. Vous devez compléter le code HTML pour lier l'événement de `click` de chaque bouton à la bonne fonction. L'action de jouer une chanson vous est donnée à titre d'exemple. Vous devez compléter le code dans `timeline-container` pour lier les bons attributs pour le temps en cours (`timeline-current`) et la barre de progrès (`timeline`).

Conseils pour la réalisation du travail pratique

1. Commencez par la configuration et l'implémentation de la communication avec la base de données MongoDB.
2. Lisez bien les tests fournis pour vous aider à implémenter les fonctionnalités demandées. Lancez les tests souvent pour vous valider votre implémentation.
3. Si votre implémentation est correcte, votre site web de TP4 devra pouvoir fonctionner avec votre serveur de TP5 de manière transparente.
4. Consultez les exemples de MongoDB sur le GitHub du cours.
5. Consultez les exemples de composantes React sur cet exemple sur le GitHub du cours ainsi que cet exemple pour l'utilisation des `Context` et `Reducer`.
6. Respectez l'approche de développement réactif et fonctionnel de React. Évitez la mutabilité de l'état.

5 Remise

Voici les consignes à suivre pour la remise de ce travail pratique :

1. Le nom de votre entrepôt Git doit avoir le nom suivant : **tp5_matricule1_matricule2** avec les matricules des 2 membres de l'équipe.
2. Vous devez remettre votre code (*push*) sur la branche **master** de votre dépôt git. (pénalité de 5% si non respecté)
3. Le travail pratique doit être remis avant **23h55**, le **4 décembre**.

Aucun retard ne sera accepté pour la remise. En cas de retard, la note sera de **0**.

Le navigateur web **Google Chrome** sera utilisé pour tester votre site web. Vos serveurs (statique et dynamique) doivent être déployables avec la commande **npm start**.

6 Évaluation

Vous serez évalués sur le respect des exigences fonctionnelles de l'énoncé, ainsi que sur la qualité de votre code JS/JSX et sa structure. Le barème de correction est le suivant :

Exigences	Points
Implémentation du serveur web et site web	
Implémentation des services du serveur	6
Implémentation des composantes de pages React	5
Implémentation des autres composantes React	5
Qualité du code	
Structure du code	2
Qualité et clarté du code	2
Total	20

L'évaluation se fera à partir de la page d'accueil du site. À partir de cette page, le correcteur devrait être capable d'interagir avec les différents éléments du site.

L'évaluation des tests se fera à partir des tests unitaires dans le projet `server`. Tous les tests fournis doivent obligatoirement passer. Tous les tests unitaires additionnels écrits par vous (si lieu) doivent obligatoirement passer.

Ce travail pratique a une pondération de **7%** sur la note du cours.

7 Questions

Si vous avez des interrogations concernant ce travail pratique, vous pouvez poser vos questions sur Discord ou contacter votre chargé de laboratoire.