



**POLYTECHNIQUE
MONTRÉAL**

UNIVERSITÉ
D'INGÉNIERIE

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

Travail pratique 5

Chargés de laboratoire:

Minh-Tri Do

Gourdigou Junior Patrice Kolani

Hiver 2022

Département de génie informatique et génie logiciel

1 Objectifs

Le but de ce travail pratique est de vous familiariser avec le développement d’une application *full stack* : système complet qui comporte un site web, un serveur web et base de données. Pour ce faire, vous utiliserez *ReactJS* pour le site web et la base de données MongoDB à partir d’un serveur NodeJS/Express pour le côté serveur. Cet ensemble de technologies est souvent 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 des différents travaux pratiques 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 votre propre serveur dynamique en utilisant l’environnement d’exécution NodeJS et la librairie Express pour assurer la distribution des fichiers statiques et des données utilisés par votre site. Vous étiez donc en présence d’un site web fonctionnel dont les informations pouvaient être partagées entre plusieurs utilisateurs ou navigateurs.

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 entre autres de la librairie React qui est basée sur l’implémentation de composants 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.

Il est intéressant de noter que lors du développement d’une application React, il est courant d’utiliser l’outil *Webpack* qui est fourni par défaut à la création d’un projet React pour assurer la transpilation en JS/HTML natifs ainsi que la distribution des fichiers statiques afin de voir en direct les changements apportés au code à l’aide d’un serveur de développement. Ainsi, lors de votre travail, il vous faudra démarrer à la fois un serveur statique qui sert votre application React ainsi que votre serveur dynamique NodeJS qui ne sera plus en charge de servir les ressources statiques du site. Plus de détails à cet effet vous sont fournis dans le README du projet.

Lors du dernier travail pratique, vous avez assuré la persistance de vos données grâce à des fichiers JSON conservés sur votre serveur. Cependant, comparativement à cette méthode, l'utilisation d'une base de données comporte une multitude d'avantages et devrait être priorisée dans la majorité des cas. Pour compléter votre application React et 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 par conséquent été modifié pour assurer la liaison entre votre site web et votre base de données.

Les 2 classes qui gèrent des recettes et les contacts du TP4 ont été renommées pour des *Services* pour mettre en évidence l'architecture orientée services de votre système. La soumission d'un nouveau commentaire à travers une requête `POST` ne fait plus de redirection du côté client : c'est au client qui fait la requête à le gérer. Ceci permet donc de découpler les services du serveur des clients qui interagissent avec.

3 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` dans l'entrepôt fourni pour le TP afin de créer votre instance MongoDB sur le service *Cloud Atlas* qui hébergera votre base de données en ligne.

Par la suite, vous aurez 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, mais vous devez compléter le reste pour retrouver l'ensemble des fonctionnalités de votre projet déjà présentes dans votre TP4.

3.1 Syntaxe JSX et serveur de développement

La particularité de la librairie React est l'utilisation de la syntaxe JSX qui est utilisée pour aider avec l'approche déclarative de développement et qui mélange l'utilisation de JS et HTML dans le même code source. Contrairement à JS et HTML, JSX n'est pas nativement 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*. React-scripts est en réalité un ensemble d'outils qui simplifient le développement de projets React. React-scripts utilise des bibliothèques tels que `Babel` pour la transpilation du code et `Webpack` pour la minification et le déploiement d'un serveur statique de développement, similaire à l'outil *lite-server* que vous avez utilisé dans les 3 premiers TPs. 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. 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:5000".

Note : le nom **localhost** est un raccourci pour l'adresse IP **127.0.0.1**. Le lancement du serveur *lite-server* affichera les adresses auxquelles le serveur est accessible.

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 la base de données MongoDB. Les tests unitaires utilisent la bibliothèque *Jest* que vous avez utilisée tout au long de la session. La bibliothèque *MongoMemoryServer* est utilisée pour remplacer une vraie instance de MongoDB par une instance contrôlée pour les tests. **Note :** Il est très important de se connecter à une base de données locale et non à votre instance MongoDB déployée en mode production pendant vos tests.

Pour lancer les suites de tests, vous n'avez qu'à aller à la racine du répertoire `server` avec un terminal et de lancer 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 pour vous aider avec le résultat final attendu. Vous pouvez ajouter vos propres tests si vous considérez que c'est pertinent.

3.3 Éléments à réaliser

La première étape consiste à implémenter les fonctionnalités nécessaires pour rendre la communication entre votre serveur et votre base de données fonctionnelle. Avant de débiter, lisez les tests fournis pour comprendre comment votre système doit fonctionner.

Rappel : Une instance MongoDB peut contenir plusieurs collections. Dans le cadre de ce TP, on vous demande de placer les différentes données dans 2 collections séparées nommées *recipes* pour les recettes et *contacts* pour les contacts respectivement.

La deuxième étape consiste à compléter le code fourni pour terminer la transition de votre site web vers un projet utilisant la librairie React. Notez que seulement les composantes fonctionnelles seront utilisées dans le cadre de ce travail.

3.4 Communication avec la base de données

Afin de mieux s'accorder avec une architecture orientée services (*SOA*), certaines des classes du serveur du TP4 ont été renommées, notamment `ContactsService` et `RecipesService`. La classe `FileSystemManager` a été remplacée par `DatabaseService` puisque la gestion des données a désormais lieu à travers une base de données 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.

3.4.1 DatabaseService

Les 4 constantes au début du fichier contiennent les informations de connexion à votre instance MongoDB. Vous devez les remplacer avec les bonnes valeurs de votre instance.

Important : suivez le guide du cours pour bien configurer votre instance MongoDB.

Vous devez implémenter la fonction `populateDb` qui permet de remplir une collection avec des données seulement si la collection est vide. Cette fonction est exécutée lors du lancement du serveur et remplit la base de données avec les valeurs des 2 fichiers JSON présents dans le répertoire `data` du projet. **Attention :** si les collections de la base de données contiennent déjà des valeurs, aucune modification ne devra avoir lieu.

3.4.2 ContactsService

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

Vous devez implémenter les fonctions suivantes de la classe : `getAllContacts`, `addNewContact` et `deleteContactById`. Consultez les en-têtes des fonctions ainsi que les tests présents dans `contacts.service.test.js` pour bien comprendre le fonctionnement des méthodes.

Notez que les méthodes sont déclarées avec le mot clé `async`. Vous avez le choix d'utiliser la syntaxe `async/await` ou `Promise`, mais soyez consistants dans l'ensemble de votre travail.

3.4.3 RecipesService

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

Vous devez implémenter les fonctions suivantes de la classe : `getAllRecipes`, `getRecipeById`, `getRecipesByIngredient`, `addNewRecipe` et `deleteRecipeById`. Consultez les en-têtes des fonctions ainsi que les tests présents dans `recipes.service.test.js` pour bien comprendre le fonctionnement des méthodes.

Notez que les méthodes sont déclarées avec le mot clé `async`. Vous avez le choix d'utiliser la syntaxe `async/await` ou `Promise`, mais soyez consistants dans l'ensemble de votre travail.

3.5 Implémentation du site web avec la librairie React

Vous devez compléter le code fourni qui permet de créer votre site web en utilisant la librairie React. Contrairement aux TPs précédents 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.

Le répertoire `services` contient du code JS qui est utilisé pour faire les appels HTTP ainsi que la gestion des 2 formulaires du projet : ajout de recette et soumission d'un contact. Comme le serveur n'est plus responsable de rediriger le client après une soumission de contact, cette logique a été déplacée du côté client, dans le composant `Contact`.

3.5.1 Composantes de pages

Le répertoire `screens` contient l'ensemble des composantes React qui définissent les différentes "pages" de votre site web et sont similaires aux fichiers HTML des TPs précédents.

Les composantes `MainPage` et `Error` vous sont fournies et vous n'avez pas à les modifier.

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.

Admin : vous devez compléter la récupération des recettes et contacts dans la fonction `useEffect` et vous devez compléter la fonction `renderContent`.

AddRecipe : vous devez compléter la fonction `addStep` ainsi que l'utilisation des valeurs de la variable `recipeInfos` dans le HTML généré. Vous devez également ajouter le composant `StepForm` pour chaque étape de la recette.

Contact : vous devez gérer l'utilisation des valeurs de la variable `contactInfos`, l'implémentation de la fonction `handleChange` ainsi que l'événement `onChange`.

Recipe : vous devez compléter la récupération de la recette dans la fonction `useEffect` et vous devez intégrer les composantes `CheckList` et `StepCard` dans le HTML généré.

Recipes : vous devez compléter les fonctions `searchByIngredient` et `toggleExactMatch`. Vous devez également intégrer les composantes `IngredientSearchBar` et `RecipeCard` en leur fournissant les bonnes propriétés.

3.6 Autres composantes React

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

Les composantes `CategoryButtonGroup`, `CheckList`, `PageFooter` et `StepCard` vous sont fournies et vous n'avez pas à les modifier sauf pour ajouter vos noms dans `PageFooter`.

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.

AdminContact : vous devez gérer l'événement `onClick` qui supprime un contact en fonction de son id.

AdminRecipe : vous devez gérer l'événement `onClick` qui supprime une recette en fonction de son id.

IngredientSearchBar : vous devez gérer l'événement `onSubmit` du formulaire ainsi que l'événement `onChange` qui modifie l'état de la variable `exactMath`.

NavBar : vous devez ajouter les bons liens vers les pages `/recipes`, `/add_recipe` et `/contact` avec les bons titres.

RecipeCard : vous devez compléter la composante en configurant le lien dynamique en fonction de l'attribut `id` de la recette.

StepForm : vous devez compléter le code HTML pour l'input de description de l'étape ainsi que le code HTML pour l'input de l'image de l'étape.

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 communiquer avec votre serveur de TP5 de manière transparente, sauf pour la redirection lors de la soumission d'un contact.
 4. Consultez les exemples de MongoDB sur le [GitHub du cours](#).
 5. Respectez la convention de codage établie par *ESLint*. Utilisez la commande `npm run lint` pour valider cet aspect.
 6. Consultez les exemples de composantes React sur le [GitHub du cours](#).
 7. Respectez l'approche de développement réactif et fonctionnel de React. Évitez la mutabilité de l'état.
-

4 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 **19 avril**.

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 (*client* et *server*) doivent être déployables avec la commande **npm start**.

5 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. Plus précisément, 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	6
Implémentation des autres composantes React	4
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'évaluations 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 **6%** sur la note du cours.

6 Questions

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

Annexe

Contact

Contactez *Les recettes de PolyEats* via le formulaire suivant:

Envoyez-nous un message

Votre nom:

Votre adresse courriel:

Votre message

Envoyez le message

FIGURE 1 – Formulaire pour un nouveau contact


Catégories

Toutes les recettes	Recettes végétariennes	Recettes méditerranéennes	Recettes ketos
---------------------	------------------------	---------------------------	----------------

Recettes


2 recettes

Pizza Hawaïienne au bacon



🕒 30 minutes

Risotto aux champignons sauvages



🕒 30 minutes

FIGURE 2 – Filtrage de recettes par catégorie

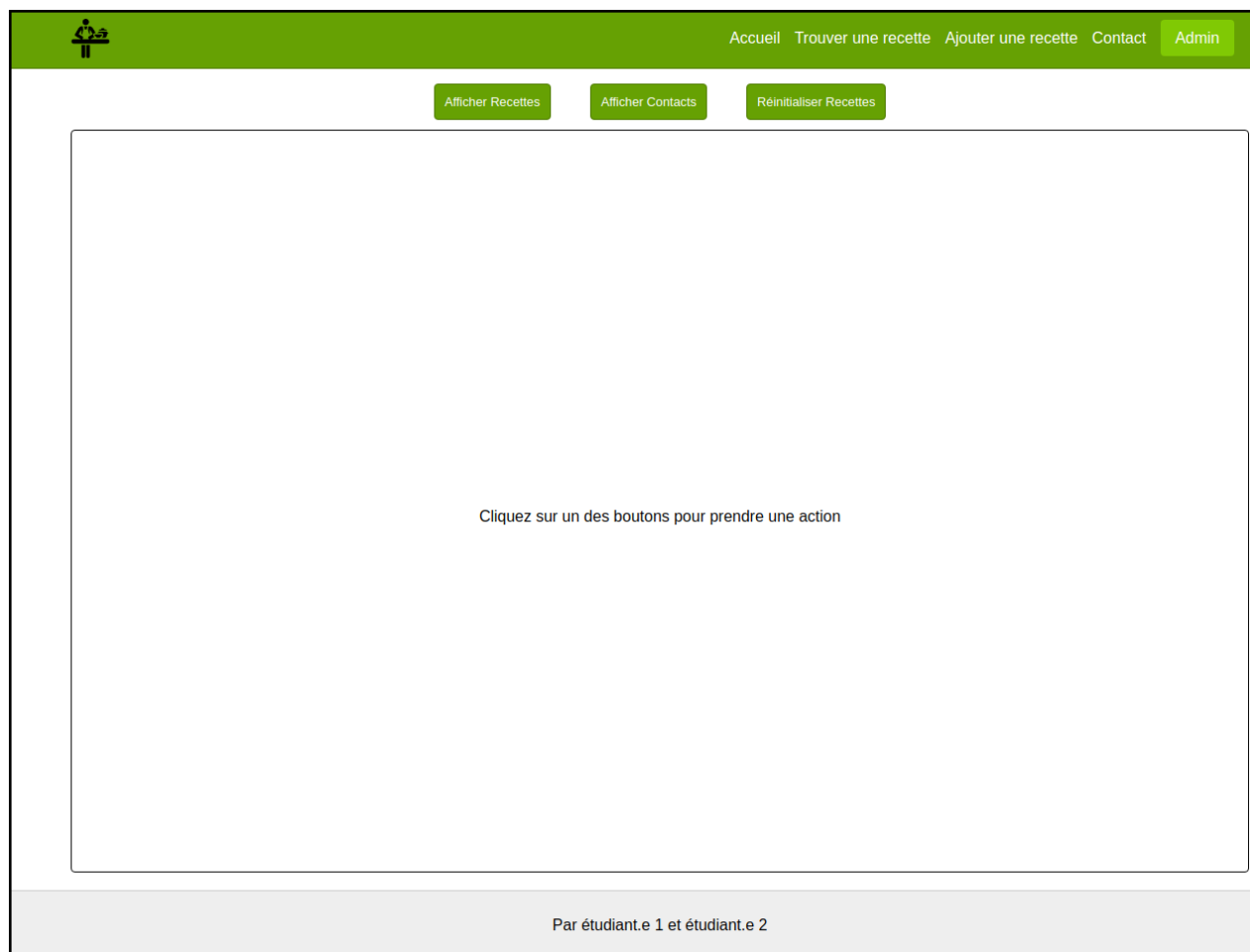


FIGURE 3 – Page administrative

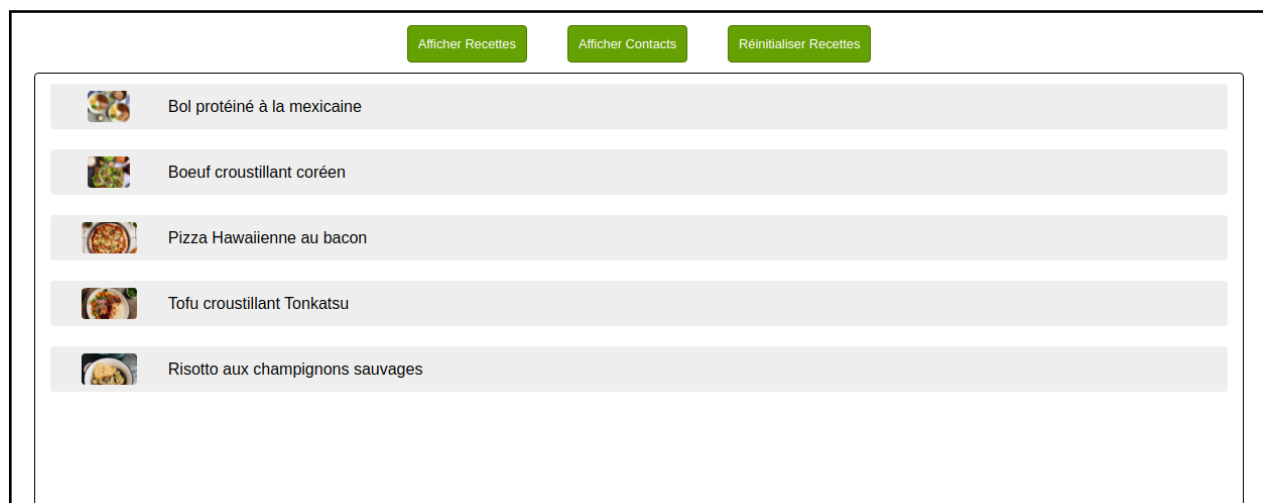


FIGURE 4 – Affichage des recettes sur la page d'administration

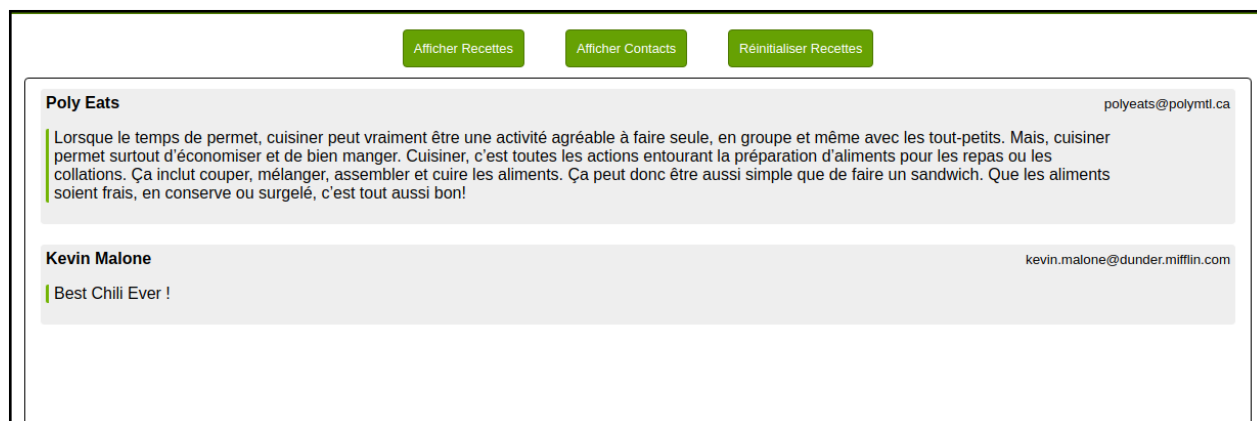


FIGURE 5 – Affichage des messages des contacts sur la page d'administration