Nom :
Prénom :
Compte examen / Clef USB :

Institut Paul Lambin — Session de janvier 2019

Examen de l'UE : Développement Web – Questions spéciales

Titulaire : Leleux Laurent
Bloc : 3BIN

Date et heure : 18/01/2019 à 8h30
Locaux : 017, 019, 025, 026
Durée de l'examen : 3h, pas de sortie durant les 60 premières minutes.

Nombre de feuilles distribuées, y compris la page de garde : 1

Consignes : sur machine, à cours fermé.

Préambule

Vous avez à votre disposition :

- Un dossier `Documentation` contenant la documentation d'Express, Node.js et le driver MongoDB.
- Un dossier contenant les sources d'un projet de base qu'il vous faudra améliorer. Il contient déjà le dossier `node_modules` avec toutes les dépendances, donc inutile de faire un `npm install`.
- Un dossier `mongodb` qui contient le serveur de base de données que vous pouvez exécuter.

Lisez attentivement le `README` qui se trouve dans les sources du projet pour savoir comment démarrer.

1. Insertion d'un user au démarrage

Au démarrage de l'application, vérifiez s'il n'existe pas déjà un user dans la base de données. Si ce n'est pas le cas, insérez-en un.

Consultez le README pour connaître la structure des documents.

N'hésitez pas à créer des fonctions supplémentaires dans le fichier `db.js` pour vous aider.

2. Authentification

Dans le client, il y a une checkbox en haut à droite dans le menu. Lorsqu'elle est cochée, on considère qu'on est authentifié, lorsqu'elle n'est pas cochée, on considère qu'on n'est pas authentifié.

Le système d'authentification est très simple ici :

- Il n'y a pas de méthode de login, la checkbox permet de savoir si on est authentifié ou non.
- Il n'y a pas de token, mais juste un header `X-AUTHENTIFIED` qui à la valeur « `true` » lorsque l'utilisateur est authentifié.
- Si le serveur reçoit ce header à « `true` », il y a un middleware qui va rechercher dans la DB le premier user venu, et le mettre dans la variable `request.user`.
- Le serveur propose une route `/users/me` qui renvoit le user qui à été mis dans `request.user` par le middleware.
- Le client fait appel à cette route pour afficher le nom de l'utilisateur connecté sur le dashboard.

Implémentez ce mécanisme en tenant compte des parties déjà implémentées (checkbox, header...).

Le fichier `utils/api.js` contient déjà beaucoup de code, c'est la partie REACT qu'il faut surtout implémenter : le fait de cocher ou non la case doit inscrire une valeur dans le `localStorage` qui sera ensuite lue par `utils/api.js`, elle doit aussi inscrire la valeur dans le state du composant `Navigation` qui doit donc devenir un composant de type classe (plutôt qu'un composant fonctionnel).

3. Tâches

Dans l'onglet tâches de l'application fournie, développez une liste de tâches simple :

- liste de toutes les tâches
- ajout d'une tâche : le formulaire doit être sur une autre page. Il faut employer React-Router pour afficher le formulaire et pour revenir sur la liste après avoir ajouté une nouvelle tâche.
- suppression d'une tâche

Ne faites donc pas la modification d'une tâche.

Côté front-end, complétez notamment le fichier `react/services/tasks.js` qui vous est fourni. La mise à page n'a pas d'importance. Toute la gestion du `state` se fait dans le `container`, les autres composants reçoivent ce qu'ils doivent via des `props`.

Côté backend, implémentez toutes les routes de CRUD habituelles, même pour les opérations qui ne seront pas utilisées par le front-end actuellement. Il n'est pas nécessaire de gérer l'authentification. Consultez le README pour connaître la structure des documents.