Guide d’étapes clés : Utilisez une API pour un compte utilisateur bancaire avec React

Comment utiliser ce document ?

Sur cette page, vous trouverez un exemple de découpage en étapes pour réaliser votre projet. Vous y trouverez :

- des recommandations pour réussir chaque étape ;
- les problème récurrents et points de vigilance à garder en tête ;
- une estimation de votre avancement sur l’ensemble du projet.

Ce découpage est simplement une suggestion pour vous organiser. Vous n’êtes pas obligé de compléter les étapes dans l’ordre.

Gardez en tête que votre progression sur les étapes n’est qu’une estimation, et sera différente selon votre vitesse de progression.

Étape 0 : Installez l'application en local

5 % de progression

Une fois cette étape terminée, je devrais avoir :

- Le back-end du site fonctionnel.
- Testé les routes de login et récupération du profil utilisateur.

Recommandations :

- Commencez par forker le repo original du projet.
- Suivez les instructions du fichier ReadMe.md.
- Une fois que le projet est lancé et que la base de données a été remplie, vous pouvez tester votre application. Pour cela, vous pouvez ouvrir la documentation de l'API à l'adresse : http://localhost:3001/api-docs. Vous pouvez tester les routes dans Postman pour vous assurer de bien récupérer les valeurs attendues.

Points de vigilance :

- Si vous rencontrez des difficultés lors de l'installation de l'application ou du chargement des données en base, vérifiez que vous avez des versions de Node ou de mongoDB compatibles installées sur votre ordinateur.

Ressources :

- En cas de difficultés avec l'installation de l'application, consultez le site de Node.js et le site de MongoDB.
- Pour vous faciliter la gestion de versions de Node, vous pouvez utiliser nvm (pour les utilisateurs de Windows, bien lire la section Important Notes)
- Découvrez pourquoi utiliser des API REST dans nos projets web avec la première partie du cours *Adoptez les API REST dans vos projets web*.

Tâche 1 : Authentification des utilisateurs

Étape 1 : Mettez en place l'application et intégrez les maquettes

25 % de progression

Durant cette première étape, nous allons créer notre application au travers de Create React App, et nous allons transformer le code que nous avons en HTML en React.

Avant de démarrer cette étape, je dois avoir :

- Installé le back-end de l'application, et testé la route de connexion (login) pour m'assurer du bon fonctionnement du back-end.

Une fois cette étape terminée, je devrais avoir :

- L'application React créée.
- Les différentes routes gérées par React Router.
- Les pages HTML/CSS de l'application intégrées avec React.

Recommandations :

- Si vous souhaitez voir un exemple de l'implémentation de Redux, vous pouvez suivre les recommandations de la documentation de Redux, et installer votre application en vous basant sur le template Redux. Dans ce cas, prenez le temps d'analyser le code présent. Vous verrez qu'une petite application de compteur est fournie, utilisant Redux et Redux Toolkit.
- Avant de commencer à intégrer les différents composants, installez React Router pour gérer la navigation dans l'application.
- Découpez les différentes pages en composants pertinents.

Points de vigilance :

- N'oubliez pas d'oPensez à bien découper vos composants ; pour cela, demandez-vous quels éléments de l'interface peuvent être réutilisés.

Ressources :

- La section "Transformez votre application en single page application avec React Router" du cours "Créez une application React Complète".

Étape 2 : Mettez en place le state global avec Redux et la connection de l'utilisateur

45 % de progression

Avant de démarrer cette étape, je dois avoir :

- Intégré le contenu des différentes pages avec React et React Router.

Une fois cette étape terminée, je devrais avoir :

- La gestion de la connexion utilisateur fonctionnelle dans mon application, et les premiers éléments présents dans le store de Redux.

Recommandations :

- Avant de commencer à écrire vos appels API, posez-vous ces questions :
  - Qu'est-ce que je cherche à récupérer à travers cet appel ?
  - Comment devrais-je gérer une réponse correcte de mon API ?
  - Que faire en cas d'erreur (par exemple sur les identifiants de l'utilisateur) ?
  - Est-ce que j'aurai besoin de garder les informations de la réponse pour pouvoir naviguer de manière fluide dans l'application, une fois connecté ?
- Les réponses à ces questions devraient vous aider à bien choisir quelles informations garder dans le store de Redux.

Points de vigilance :

- On ne devrait pouvoir accéder à la page de profil de l'utilisateur qu'une fois connecté.
- Certaines informations du profil de l'utilisateur connecté s'affichent à différents endroits de l'application ; ces éléments devront être des composants différents, néanmoins ils devront être alimentés par la même source de données.
- Pensez à gérer la déconnexion de l'utilisateur, et donc la disparition de ses informations dans le store de l'application.

Ressources :

- La section “Utilisez le thunk middleware" du cours “Utilisez le state manager Redux pour gérer l’état de vos applications”, pour rappel de la bonne gestion des appels réseaux avec Redux.
- La section “Unifiez actions et reducers avec les slices” du même cours, pour bien écrire la logique de Redux avec Redux Toolkit.

Étape 3 : Modifiez le nom d’utilisateur via le formulaire

65 % de progression

L’application est maintenant sécurisée, et nous pouvons naviguer entre les différentes pages sans être déconnecté. Il est temps maintenant de passer à une feature clé de l’application Argent Bank : la modification du nom d’utilisateur.

Avant de démarrer cette étape, je dois avoir :

- Sécurisé la connexion de l’utilisateur.

Une fois cette étape terminée, je devrais avoir :

- La possibilité de mettre à jour mon username.

Recommandations :

- Commencez par créer le formulaire ; puis, afin de traiter au mieux l’appel à l’API pour réaliser l’actualisation, posez-vous ces questions :
    - Ai-je besoin que les modifications persistent ?
    - Une fois les modifications enregistrées, où doit se répercuter le changement de l’affichage pour que l’ensemble de la page soit cohérent ?

Points de vigilance :

- Pensez bien au cycle de vie des composants lorsque ceux-ci utilisent le store de Redux.

Tâche 2 : Gestion des transactions

Voilà, l'application est fonctionnelle. Félicitations ! Nous allons maintenant travailler sur la gestion des transactions. Il s'agira de spécifier les endpoints d'API nécessaires pour étendre les fonctionnalités du tableau de bord.

Étape 4 : *Analyse des maquettes*

75 % de progression

Avant de démarrer cette étape, je dois avoir :

- L'application React Redux complètement fonctionnelle.

Une fois cette étape terminée, je devrais avoir :

- Une liste des endpoints nécessaires au bon fonctionnement de l'application.

Recommandations :

Lors de l'analyse des maquettes, posez-vous ces questions :

- Quelles données cherche-t-on à récupérer pour les transactions ?
- Que peut-on ajouter, modifier ou supprimer, et quelles actions cela implique-t-il au niveau de l'API ?

Cela vous aidera à pouvoir lister toutes les actions nécessaires au bon fonctionnement de l'API.

Points de vigilance :

Vous n'êtes pas forcément obligé d'utiliser toutes les méthodes du CRUD (Create Read Update Delete) dans votre spécification. Il sera cependant important que vous soyez prêt à justifier vos choix durant la soutenance.

Ressources :

- Découvrez les grands axes de la création d'API REST avec la partie 3 du cours "Adoptez les API REST pour vos projets web".

Étape 5 : Écrivez les différents endpoints dans le fichier Swagger

90 % de progression

C’est bon, vous avez défini les différents endpoints de votre API ? Il est temps de les intégrer dans le document Swagger de l’application.

Avant de démarrer cette étape, je dois avoir :

- La liste de mes endpoints pour les transactions.

Une fois cette étape terminée, je devrais avoir :

- Le document Swagger avec les opérations de transaction.

Recommandations :
Pour écrire les routes de l’API, utilisez la base qui vous est donnée avec les utilisateurs, cela vous évitera de commencer de zéro.

Points de vigilance :
Le document actuel suit la nomenclature Swagger 2.0, choisissez la bonne documentation pour poursuivre l’intégration des routes.

Ressources :

- La documentation Swagger pour vous aider dans la rédaction des routes.

Étape 6 : *Relecture et préparation des livrables*

100 % de progression

Voilà, le travail est fini, il est temps de faire une relecture de l’ensemble, et de préparer les livrables du projet.

Avant de démarrer cette étape, je dois avoir :

- L’application React Redux finalisée.
- Le document Swagger rédigé.

Une fois cette étape terminée, je devrais avoir :

- Les livrables déposés sur la plateforme.

Recommandations :

- Lancez le projet et testez les fonctionnalités demandées pour être sûr que tout fonctionne correctement.
- Relisez votre document Swagger. Assurez-vous d’avoir défini toutes les routes permettant de réaliser les fonctionnalités demandées.

Projet terminé !