Evaluation de React - Application POKEDEX

Vous êtes chargé de réaliser une application web de type POKEDEX en utilisant React pour une entreprise de développement de jeux vidéo.

Vous allez être guidé tout au long de cette évaluation pour réaliser cette application. Chaque étape est détaillée et vous devrez réaliser les tâches demandées. Plus vous avancerez dans les étapes, plus les tâches seront difficiles et demanderont de la réflexion. Et moins de détails vous seront donnés.

Il est important de réaliser toutes les étapes dans l'ordre pour ne pas être perdu. Et idéalement de faire un commit à chaque étape pour garder une trace de votre progression. (Vous pouvez aussi faire un commit à chaque tâche si vous le souhaitez)

Étape 1 : Initialisation du projet

1. Créez un nouveau projet React nommé `pokedex`.
2. Dans le composant `App`, affichez le titre de l'application "POKEDEX" dans une balise `h1`.

Étape 2 : Création d'un composant `PresentationPokedex`

1. Créez un composant `PresentationPokedex` qui contiendra :
    - Le `h1` "POKEDEX"
    - Un paragraphe de présentation de l'application :
        - "Découvrez le monde fascinant des Pokémon avec notre Pokédex ultime! Que vous soyez un Maître Pokémon en devenir ou simplement curieux de découvrir chaque créature unique, notre base de données complète est votre porte d'entrée vers une aventure extraordinaire. Explorez les caractéristiques, les forces, et les faiblesses de chaque Pokémon, plongez dans leur univers, et devenez l'expert ultime."
    - Une section présentant les fonctionnalités de l'application :
        - Recherche Instantanée par Nom: Ne perdez plus jamais de temps à chercher votre Pokémon préféré. Entrez simplement son nom pour accéder instantanément à une mine d'informations.
        - Fiches Détaillées: Chaque Pokémon est unique, et notre Pokedex célèbre cette diversité.
        - Toujours Plus à Découvrir: Nous nous engageons à enrichir constamment le Pokedex avec de nouvelles fonctionnalités passionnantes. Restez à l'affût!

    - Un formulaire d'inscription a la newsletter :
        - Un champ de saisie pour l'adresse email
        - Un bouton "S'inscrire"

2. Importez le composant `PresentationPokedex` dans le composant `App` et affichez-le.
3. Refactorisez le composant `PresentationPokedex` pour qu'il soit plus lisible et que chaque élément soit dans un composant dédié.
4. Ajoutez un style CSS pour le composant `PresentationPokedex`. Soit avec tailwindcss, soit avec un module CSS par composant.

Étape 3 : Création d'un composant `PokemonStarters`

1. Créez un composant `PokemonStarters` qui contiendra :
    - Un titre `h2` "Starters Pokémon"
    - Un titre `h3` pour chaque génération de Pokemon, contenant le nom de la génération :
        - "Rouge et Bleu"
        - "Or et Argent"
        - "Rubis et Saphir"
    - Pour chaque génération, affichez les 3 starters de la génération avec leur nom et leur image:
        - Bulbizarre, Salamèche, Carapuce
        - Germignon, Héricendre, Kaiminus
        - Arcko, Poussifeu, Gobou

Vous trouverez un jeu de données pour les starters dans starters.js.

Vous pouvez importer ce fichier dans le composant `PokemonStarters` pour afficher les starters ou bien copier-coller les données directement dans le composant.

Conseils :

- Utilisez un `map` pour afficher chaque génération.
- Utilisez un `map` pour afficher chaque pokemon de chaque génération.
- Regardez la structure des données pour savoir comment les afficher.

2. Importez le composant `PokemonStarters` dans le composant `App` et affichez-le.
3. Refactorisez le composant `PokemonStarters` pour qu'il soit plus lisible : le plus simple est de créer un composant `PokemonCard` pour chaque Pokemon.
4. Ajoutez un style CSS pour le composant `PokemonStarters`. Soit avec tailwindcss, soit avec un module CSS par composant.

Étape 4 : Création d'un composant `PokemonList`

Le but de ce composant est d'afficher la liste de la totalité des pokemons d'un coup.

Bien sur, ce n'est pas très pratique pour l'utilisateur, mais c'est un bon exercice pour vous.

1. Créez un composant `PokemonList` qui ira sur l'url `https://pokebuildapi.fr/api/v1/pokemon` pour récupérer la liste de tous les pokemons.
2. Affichez la liste des pokemons dans le composant `PokemonList` avec leur nom et leur image.
3. Importez le composant `PokemonList` dans le composant `App` et affichez-le.
4. Refactorisez le composant `PokemonList` pour qu'il soit plus lisible : le plus simple est de réutiliser le composant `PokemonCard` pour chaque Pokemon.
5. Ajoutez un style CSS pour le composant `PokemonList`. Soit avec tailwindcss, soit avec un module CSS par composant.

Bonus : Afficher un loader pendant le chargement des données.

Étape 5 : Création d'un composant `PokemonSearch`

Le but de ce composant est de permettre à l'utilisateur de rechercher un pokemon par son nom.

1. Créez un composant `PokemonSearch` qui contiendra :
    - Un champ de saisie pour le nom du pokemon
    - Un bouton "Rechercher"
2. Lorsque l'utilisateur entre le nom d'un pokemon (ou juste quelques charactères) et clique sur le bouton "Rechercher", recherchez le pokemon correspondant dans la liste des pokemons et affichez-le.
3. Importez le composant `PokemonSearch` dans le composant `App` et affichez-le.

4. Refactorisez le composant `PokemonSearch` pour qu'il soit plus lisible : par exemple, vous pouvez extraire la logique de recherche dans une fonction dédiée.
5. Ajoutez un style CSS pour le composant `PokemonSearch`. Soit avec tailwindcss, soit avec un module CSS par composant.

Tips : vous pouvez réutiliser la requête de l'étape précédente pour récupérer la liste des pokemons.

Étape 6 : Création d'un composant `PokemonTeam` et de la logique de sélection

Le but de ce composant est de permettre à l'utilisateur de sélectionner ses pokemons préférés pour former une équipe.

1. Créez un composant `PokemonTeam` qui contiendra :
    - Un titre `h2` "Mon équipe Pokémon"
    - La liste des pokemons sélectionnés par l'utilisateur
    - S'il n'y a pas de pokemons sélectionnés, affichez un message "Aucun pokemon sélectionné"
2. Ajoutez un bouton "Ajouter à mon équipe" dans le composant `PokemonCard` pour chaque pokemon.
3. Lorsque l'utilisateur clique sur le bouton "Ajouter à mon équipe", ajoutez le pokemon à la liste des pokemons sélectionnés.
4. Importez le composant `PokemonTeam` dans le composant `App` et affichez-le.
5. Refactorisez le composant `PokemonTeam` pour qu'il soit plus lisible : par exemple, vous pouvez extraire la logique de sélection dans une fonction dédiée.
6. Ajoutez un style CSS pour le composant `PokemonTeam`. Soit avec tailwindcss, soit avec un module CSS par composant.

Bonus 1 : Ajoutez un bouton "Retirer de mon équipe" pour chaque pokemon dans la liste des pokemons sélectionnés. Bonus 2 : Dans le composant `PokemonCard`, changez le bouton "Ajouter à mon équipe" en "Retirer de mon équipe" si le pokemon est déjà dans la liste des pokemons sélectionnés. Bonus 3 : Ajoutez un bouton "Vider mon équipe" pour vider la liste des pokemons sélectionnés. Bonus 4 : Bloquer l'ajout d'un pokemon si la liste des pokemons sélectionnés est déjà pleine (6 pokemons maximum).

Tips :

- Vous pouvez utiliser le state pour stocker la liste des pokemons sélectionnés.
- Vous pouvez utiliser la méthode `filter` sur le tableau de la totalité des pokemons pour ne garder que les pokemons qui sont sélectionnés.

Etape 9 : Sauvegarde local de l'équipe de pokemons

Le but de cette étape est de sauvegarder localement la liste des pokemons sélectionnés par l'utilisateur.

1. Utilisez `localStorage` pour sauvegarder la liste des pokemons sélectionnés.
2. Utilisez `localStorage` pour récupérer la liste des pokemons sélectionnés au chargement de la page.
3. Utilisez `localStorage` pour mettre à jour la liste des pokemons sélectionnés lorsqu'elle est modifiée.

Tips :

- La sauvegarde et la récupération des pokemons sur le localstorage est un effet de bord de votre application !
- Vous pouvez utiliser `JSON.stringify` pour sauvegarder un objet dans `localStorage` et `JSON.parse` pour le récupérer.

Etape 8 : Mise en place du routing et de la navigation

Le but de cette étape est de mettre en place le routing pour naviguer entre les différentes pages de l'application.

1. Installez `react-router-dom` dans votre projet.
2. Créez un composant `Header` qui contiendra :
    - Un titre `h1` "POKEDEX"
    - Un menu de navigation avec 3 liens :
        - "Accueil" qui redirige vers la page d'accueil
        - "Liste des Pokemons" qui redirige vers la page de la liste des pokemons
        - "Mon équipe" qui redirige vers la page de l'équipe de pokemons
        - "Recherche" qui redirige vers la page de recherche de pokemons

3. Créez les composants `Home`, `PokemonListPage`, `PokemonSearchPage` et `PokemonTeamPage` qui contiendront respectivement les composants `PresentationPokedex`, `PokemonList`, `PokemonSearch` et `PokemonTeam`.

4. Mettez en place le routing dans le composant `main.jsx` pour afficher les composants `Header` et `Home` par défaut, et les composants `PokemonListPage`, `PokemonSearchPage` et `PokemonTeamPage` en fonction de l'url.

5. Ajoutez un style CSS pour le composant `Header`. Soit avec tailwindcss, soit avec un module CSS par composant.

Bonus : Ajoutez un style CSS pour le composant `Header` pour qu'il soit fixé en haut de la page et qu'il soit toujours visible.

Etape 8 : Mise en place de la gestion de l'état global

Le but de cette étape est de mettre en place un état global pour stocker les données de l'application.

1. Utilisez le contexte pour stocker les données de l'application.

2. Stockez les données de la liste des pokemons et de la liste des pokemons sélectionnés dans le contexte.

3. Utilisez le contexte dans les composants `PokemonList`, `PokemonSearch` et `PokemonTeam` pour récupérer les données de la liste des pokemons et de la liste des pokemons sélectionnés.

4. Utilisez le contexte dans les composants `PokemonList`, `PokemonSearch` et `PokemonTeam` pour mettre à jour les données de la liste des pokemons et de la liste des pokemons sélectionnés.

5. Supprimez les données de la liste des pokemons et de la liste des pokemons sélectionnés du state des composants `PokemonList`, `PokemonSearch` et `PokemonTeam`.