1. Les principes de base du framework ReactJS

Consigne :
Créez les composants de base de votre application Trello :

- Un composant `Board` qui contiendra plusieurs `List`.
- Un composant `List` qui contiendra plusieurs `Task`.
- Chaque composant React doit utiliser les props pour recevoir et afficher les données de tâches et listes.
- N'oubliez pas d'utiliser le rendu via le `map()` pour afficher plusieurs tâches dans une liste.

2. Gérer les événements et rendu conditionnel dans le JSX

Consigne :
Implémentez un système de gestion des événements et des rendus conditionnels :

- Faites un rendu conditionnel pour afficher un message "Aucune tâche" si une liste est vide.
- Les gestions d'événements seront utiles plus tard lors de l'implémentation des formulaires d'ajout de tâches. Vous les utiliserez à ce moment-là.

3. Navigation avec React et react-router-dom

Consigne :
Ajoutez une navigation entre plusieurs vues dans l'application :

npm install react-router-dom

- Créez au moins deux pages : une page d'accueil `Home` avec un aperçu des boards présents avec `Board` et une page `BoardDetails` pour afficher un seul board.
- Implémentez le routage avec `react-router-dom` pour naviguer entre les boards.
- Utilisez les paramètres de route pour identifier quel board est affiché !

4. Présentation des Hooks et des formulaires

Consigne :
Utilisez des Hooks pour gérer l'état des composants et les formulaires :

- Modifiez la gestion de l'état des composants (`Task`, `List`, …) en utilisant le Hook `useState`.
- Ajoutez un formulaire pour ajouter des tâches à une liste, en utilisant les Hooks pour gérer les champs de saisie.
- Implémentez le Hook `useEffect` pour afficher un message lorsqu'une liste est modifiée.

5. Le concept de store avec Redux

Consigne :
Intégrez Redux pour gérer l'état global de l'application :

npm install redux react-redux @reduxjs/toolkit

- Créez un store Redux pour gérer les données des boards, des listes et des tâches.
- Utilisez des actions et des reducers pour ajouter des tâches à l'état global.
- Configurez `Provider` pour que tous les composants puissent accéder au store.

6. Les tests avec Jest et React Testing Library (RTL)

Consigne :
Ajoutez des tests unitaires à votre application :.

- Écrivez des tests Jest pour vérifier que les reducers de Redux fonctionnent correctement.
- Utilisez RTL pour tester le fichier `TaskForm.jsx`.

7. Bonus

- Faites du CSS
- Faites en sorte de pouvoir ajouter des board et des listes de façon dynamique
- Mettez en place une `sauvegarde des données dans le localStorage` via Redux afin que les boards et tâches persistent après un rafraîchissement.
- Ajoutez un système de `drag-and-drop` (bibliothèque externe comme `react-beautiful-dnd` ou un événement `drag/drop` pur en JS).
- Implémentez une fonctionnalité d'`édition` des tâches (double clic pour modifier le titre d'une tâche).
- Utilisez des `Promesses` ou `async/await` pour simuler une requête API qui charge les listes initiales.
- Implémentez une redirection automatique après la création d'un nouveau board (par exemple, vers la page du board créé).