مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

Contrôle Continu N° 02 – Surveillé -

Filière : Développement Digital **(DEVOWFS201)** **Module N°204** : Développement Frontend

EFP **:** ISTA OUARZAZATE **Durée** : 3H00

PARTIE II : Utiliser les Hooks et Consommer les APIs

SUJET 01 : **Gestion des emprunts dans une bibliothèque** (20pts)

Le travail consiste à créer une application ReactJS qui gère les emprunts dans une bibliothèque en utilisant les Hooks (`useState`, `useEffect`, `useContext`) et consomme une API en format JSON pour obtenir la liste des livres disponibles.

https://gahi-said.com/apis/auteurs.php

Composant	Rôle	Lien avec les autres composants
App	Point d'entrée principal de l'application.	Contient les composants **ListLivre** et **LivresEmpruntes**. Enveloppé dans le **EmpruntProvider** pour fournir le contexte global.
ListLivre	Affiche la liste des livres disponibles en consommant l'API.	- Utilise le contexte `EmpruntContext` pour accéder aux fonctions `EmpruntLivre`. - Récupère les données de l'API avec `useEffect`.
LivresEmpruntes	Affiche les livres actuellement empruntés.	- Utilise le contexte `EmpruntContext` pour accéder à la liste des emprunts et à la fonction `returnLivre`.
EmpruntProvider	Fournit le contexte global pour gérer les emprunts à travers l'application.	- Partagé avec les composants enfants comme **ListLivre** et **LivresEmpruntes** via le contexte **EmpruntContext**.
Message	Composant pour afficher les messages d'erreur ou de confirmation (ex. "Livre emprunté avec succès").	- Utilisé dans **ListLivre** ou **LivresEmpruntes** pour informer l'utilisateur des actions réussies ou échouées.

```
├── src/
│   ├── components/            # Dossier contenant tous les composants React
│   │   ├── ListLivre.js       # Composant affichant la liste des livres
│   │   ├── LivresEmpruntes.js # Composant affichant les livres empruntés
│   │   ├── Message.js         # Composant pour afficher des messages
│   ├── context/        # Dossier pour le contexte de gestion des emprunts
│   │   ├── EmpruntContext.js #contexte fournisseur global pourles emprunts
│   ├── services/              # Dossier pour les services (API)
│   │   ├── api.js             # Service pour consommer l'API des livres
│   ├── App.js       # Composant principal, point d'entrée de l'application
│   ├── index.js               # Point d'entrée JavaScript
```

Partie 1 : Initialisation et architecture du projet (2pts)

1. **Question 1** : Initialisation du projet
   - Créez une nouvelle application ReactJS avec `create-react-app` et nommez-la `bibliotheque-react`.
   - Installez les dépendances nécessaires : `axios` pour les requêtes HTTP et toute autre bibliothèque que vous jugez utile.
2. **Question 2** : Organisation du projet
   - Créez une architecture de projet avec les dossiers suivants :
     - **components** : pour les composants React.
     - **context** : pour le contexte global de gestion des emprunts.
     - **services** : pour les appels à l'API.

Partie 2 : Consommation de l'API et affichage des livres (6pts)

API (à utiliser pour la liste des livres)

3. **Question 3** : Création du service pour l'API
   - Implémentez une fonction **`fetchLivres()`** dans un fichier `services/api.js` pour récupérer la liste des livres depuis l'API.
   - Utilisez **`axios`** ou **`fetch`** pour effectuer la requête.
4. **Question 4** : Affichage des livres
   - Créez un composant `ListLivre` qui utilise le Hook `useEffect` pour appeler `fetchLivres()` au chargement.
   - Affichez les livres sous forme de tableau ou de liste avec les informations suivantes : **titre**, **auteur**, **disponibilité**.

Partie 3 : Gestion des emprunts avec un contexte global (6pts)

5. **Question 5** : Création du contexte global
   - Créez un contexte `EmpruntContext` dans le dossier `context`.
   - Ajoutez un `EmpruntProvider` qui contient un état global pour gérer les emprunts sous forme de tableau
6. **Question 6** : Ajout de la logique d'emprunt
   - Ajoutez une fonction `EmpruntLivre(id)` dans le contexte pour ajouter un livre aux emprunts, à condition qu'il soit disponible.
   - Créez une fonction `returnLivre(id)` pour rendre un livre emprunté.

Partie 4 : Interactions avec les composants (6pts)

7. **Question 7** : Bouton d'emprunt et de retour
   - Dans le composant `ListLivre`, affichez un bouton `Emprunter` pour chaque livre disponible.
   - Ajoutez un bouton `Rendre` pour chaque livre déjà emprunté. Ces boutons doivent appeler `EmpruntLivre` et `returnLivre`.
8. **Question 8** : Affichage des emprunts
   - Créez un composant `LivresEmpruntes` pour afficher la liste des livres empruntés.
   - Utilisez `useContext` pour accéder aux emprunts depuis le contexte global.