Projet aventures alpines

Table des matières

Présentation des outils

REACT

- Bibliothèque JavaScript open-source développée et maintenue par Facebook, utilisée pour la création d'interfaces utilisateur (UI). Son principal objectif est de faciliter la création d'applications à interface utilisateur interactive et dynamique.

Distinction fondamentale entre une bibliothèque et un framework

- Elle réside dans le contrôle que vous avez sur votre application :
  - Dans un framework, comme Symfony, vous suivez une structure prédéfinie et vous utilisez les composants et les fonctionnalités fournis par le Symfony.
  - Dans une bibliothèque comme React, vous avez plus de flexibilité pour concevoir la structure de votre application et n'utilisez que les parties de la bibliothèque dont vous avez besoin.

Objectif de React

- Se concentrer principalement sur la gestion de l'interface utilisateur. Il fournit des composants réutilisables et un système de rendu efficace qui vous permet de créer des interfaces utilisateur interactives. Cela signifie que vous pouvez utiliser React avec d'autres bibliothèques et frameworks de votre choix pour gérer d'autres aspects de votre application, tels que la gestion de l'état, le routage, etc.

Axios pour les requêtes http :

- Bibliothèque JavaScript populaire qui permet d'effectuer des requêtes HTTP depuis un navigateur web ou depuis Node.js. Elle est couramment utilisée pour interagir avec des API web, envoyer des données au serveur et récupérer des données du serveur.

- **Facilité d'utilisation** :
  - Offre une syntaxe simple et cohérente pour effectuer des requêtes http
  - Prend en charge les requêtes GET, POST, PUT, DELETE et d'autres méthodes HTTP courantes.
- **Compatibilité des navigateurs** :
  - Compatible avec tous les navigateurs modernes
  - Choix fiable pour les applications web.
- **Promesses** :
  - Utilise des promesses JavaScript pour gérer les réponses aux requêtes HTTP. → Vous pouvez utiliser `then()` et `catch()` pour gérer de manière **asynchrone** les résultats réussis et les erreurs de vos requêtes. Cela rend la gestion des appels asynchrones plus propre et plus lisible que les callbacks.

- **Gestion automatique des données JSON** :
  - Convertit automatiquement les données JSON renvoyées par le serveur en objets JavaScript et inversement.
- **Interception des requêtes et des réponses** :
  - Permet d'intercepter et de modifier les requêtes et les réponses HTTP avant qu'elles n'atteignent leur destination. Cela peut être utile pour ajouter des en-têtes d'authentification, gérer les erreurs de manière globale, etc.
- **Annulation de requêtes** :
  - Prend en charge l'annulation de requêtes en permettant l'utilisation de "cancel tokens"
    - → Permet d'annuler des requêtes en cours si elles ne sont plus nécessaires
    - → Amélioration des performances et de la gestion des erreurs dans une application.
- **Protection contre les attaques CSRF** :
  - Peut-être configuré pour inclure automatiquement un en-tête CSRF lors des requêtes POST
    - → Amélioration de la sécurité contre les attaques CSRF (Cross-Site Request Forgery).

API

- Ensemble de règles et de protocoles qui permettent à différentes applications logicielles de communiquer entre elles
- Définit les méthodes et les structures de données que les développeurs peuvent utiliser pour accéder aux fonctionnalités d'un logiciel ou d'un service particulier sans avoir à comprendre les détails internes de son implémentation.

- **Communication entre logiciels** :
  - Facilitent la communication entre différentes applications logicielles, qu'elles soient exécutées sur le même appareil, sur des appareils différents ou sur des serveurs distants
  - Permettent à ces applications de demander des données, d'envoyer des données ou d'exécuter des actions à travers une interface standardisée.
- **Abstraction** :
  - Fournissent une abstraction qui masque la complexité interne d'un logiciel ou d'un service
  - Permet aux développeurs d'interagir avec le logiciel de manière simplifiée et de manière cohérente, quel que soit le logiciel sous-jacent.
- **Réutilisation de code** :
  - Permettent de réutiliser du code existant sans avoir à le réinventer
- **Sécurité** :
  - Peuvent être sécurisées en utilisant des mécanismes d'authentification et d'autorisation
    - → Garantit que seules les applications autorisées peuvent accéder aux fonctionnalités de l'API.
- **Documentation** :
  - Généralement accompagnées d'une documentation détaillée qui explique comment les utiliser. Cette documentation indique quelles requêtes peuvent

être faites, quelles données peuvent être envoyées et comment interpréter les réponses.

- **Types d'API** :
  - API web (utilisées pour communiquer via HTTP sur Internet)
  - API système (pour interagir avec le système d'exploitation)
  - API de bibliothèque (pour accéder à des fonctions spécifiques d'une bibliothèque)

JavaScript

- JavaScript est un langage de programmation de haut niveau, interprété et orienté objet, utilisé principalement pour le développement web

Node.js

- Environnement d'exécution JavaScript côté serveur open-source et basé sur le moteur JavaScript V8 de Google
- Permet d'exécuter du code JavaScript côté serveur
- Principalement utilisé pour la création de serveurs web, d'applications réseau et de scripts côté serveur
- **JavaScript côté serveur** :
  - Node.js permet d'exécuter du code JavaScript sur le serveur
    - → On peut donc utiliser le même langage (JavaScript) à la fois pour le côté client (dans le navigateur) et le côté serveur
    - → Gestion complète des applications web.
- **Non bloquant et asynchrone** :
  - Conçu pour être non bloquant et asynchrone
    - → Gestion de nombreuses connexions simultanées sans bloquer le traitement
    - → Particulièrement adapté aux applications en temps réel, telles que les chats en ligne ou les jeux multijoueurs.
- **Gestionnaire de paquets** :
  - Inclut npm (Node Package Manager), un gestionnaire de paquets très populaire.
- **Écosystème riche** :
  - Possède un vaste écosystème de packages et de modules open-source, ce qui facilite le développement d'applications en utilisant des composants pré-construits
- **Hautes performances** :
  - Grâce à son modèle de gestion des entrées/sorties asynchrone et à son moteur JavaScript V8 rapide, il offre de bonnes performances pour les applications nécessitant une gestion efficace des connexions simultanées.
- **Utilisations diverses** :
  - création varié d'applications, notamment des serveurs web, des applications de chat en temps réel, des outils de ligne de commande, des outils de construction de sites web statiques, des applications IoT (Internet des objets), etc.

Présentation du projet

Un Voyage Virtuel dans le monde de la Montagne

Aventures Alpines est une plateforme en ligne conçue spécialement pour les amoureux du sport de montagne et de l'exploration en altitude. Que vous soyez un grimpeur passionné, un randonneur tranquille ou simplement fasciné par la beauté majestueuse des montagnes, Aventures Alpines vous offre un espace pour vous informer, vous inspirer et partager vos expériences.

Caractéristiques Principales

a) **Découverte des Sports de Montagne :** Explorez une variété de sports de montagne, des escalades vertigineuses aux randonnées sereines, et découvrez leurs spécificités, défis et histoires.
b) **Articles Riches et Informatifs :** Plongez dans notre collection d'articles écrits par des passionnés et des experts du domaine. Apprenez les bases de l'alpinisme, découvrez des astuces pour la randonnée en haute altitude, et explorez les dernières tendances en matière d'équipement.
c) **Vidéos Captivantes :** Visionnez des vidéos époustouflantes mettant en scène des aventures en montagne. Des documentaires inspirants aux vidéos d'escalade palpitantes, notre bibliothèque de vidéos vous fera voyager depuis chez vous.
d) **Itinéraires de Randonnée :** Planifiez votre prochaine aventure en explorant nos itinéraires de randonnée soigneusement sélectionnés. Consultez les détails, les niveaux de difficulté et les vues panoramiques pour choisir la randonnée qui vous convient le mieux.
e) **Partagez Vos Expériences :** Partagez vos propres expériences en montagne avec la communauté Aventures Alpines. Publiez des photos, des récits et des conseils pour aider d'autres passionnés à vivre des aventures exceptionnelles.
f) **Communauté Engagée :** Rejoignez une communauté de personnes partageant les mêmes intérêts. Discutez des dernières actualités en montagne, posez des questions, et inspirez-vous des histoires des autres membres.
g) **Interface Conviviale :** Aventures Alpines est conçu avec une interface utilisateur moderne et conviviale, garantissant une expérience fluide et agréable pendant votre exploration.

Comment Commencer

a) **Inscrivez-vous ou connectez-vous :** Créez votre compte Aventures Alpines pour accéder à toutes les fonctionnalités de la plateforme.
b) **Explorez les Contenus :** Parcourez les articles, vidéos et itinéraires pour élargir vos connaissances et trouver l'inspiration pour votre prochaine aventure.
c) **Partagez Votre Passion :** Partagez vos aventures passées et vos projets futurs avec la communauté Aventures Alpines. Recevez des commentaires, des recommandations et des encouragements.

d) **Engagez-vous :** Rejoignez les discussions, posez des questions, et construisez des amitiés avec des personnes partageant les mêmes intérêts.
e) **Planifiez Votre Prochaine Expédition :** Explorez nos itinéraires de randonnée et préparez-vous à vivre des moments inoubliables en montagne.

Rejoignez Aventures Alpines dès aujourd'hui et plongez dans un monde de défis, d'exploration et de beauté majestueuse. Ensemble, nous atteindrons de nouveaux sommets !

Conception de l'application

Mission 1 Construire le MCD aventures alpines et le diagramme de classe

Compétences	• C1.1.1.2 Identifier les fonctionnalités attendues du service à produire • C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact • C1.1.2.1 Analyser les interactions entre services
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Créer** le MDC et le MLD de Aventures Alpines

Objectifs intermédiaires

- **Réinvestir** les notions de cours en base de données et POO

Outils nécessaires

- Suite Microsoft ou Libre Office
- IntelliJ IDEA, Visual Code ou Visual Studio 2022
- JS, node.js
- VirtualBox ou réel
- PostGréSQL ou Mysql

Pré-requis

- Cours de base de données et POO

Votre mission

Construire le MCD et le diagramme de classe à partir des spécifications données dans informations utiles

Informations utiles

Un visiteur référencé par un ID, consulte des informations qui peuvent être des articles, des blogs ou encore des vidéos laissées par des clients. Il peut aussi regarder les activités proposées (type loisir ou compétition) dédiées au sport, selon s'il s'agit d'une famille, de gens expérimenté ou de personnes à la recherche de sensation forte. Les sports proposés sont le ski, la randonnée ou l'escalade.

Pour le ski, on gère le nom du domaine skiable, savoir si oui ou non il y a des remontées mécaniques et le type de couleur de la piste (verte, jaune, rouge, noire).

Pour l'escalade, on gère le site, le niveau (facile moyen difficile expérimenté), le temps d'ascension.

Pour la randonnée, on gère la région, le lieu de départ, celui d'arrivée, le nombre de km, savoir la randonnée est praticable en été comme en hivers et si un guide est nécessaire. Dans le cas où la randonnée doit avoir un guide, on gère le nom et le prénom du guide.

Lorsque le visiteur est intéressé pour réserver, il devient client. Dans ce cas, on gère son nom, prénom et adresse (rue, cp, ville). La réservation se fait sur des prestations données dont on garde la date de début et de fin ainsi que le nom de la prestation.

Lorsque le client réserve, il le fait pour une date de début et de fin, pour un nombre de personnes et un prix par rapport à une prestation dont on gère le nom. Cette prestation concerne les activités décrites ci-dessus.

Mission 2 Spécifications du projet aventures alpines

Compétences	• C1.1.1.2 Identifier les fonctionnalités attendues du service à produire • C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact • C1.1.2.1 Analyser les interactions entre services
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Créer** la structure du projet Aventures Alpines

Objectifs intermédiaires

- **Réinvestir** les notions de cours en base de données et POO

Outils nécessaires

- Suite Microsoft ou Libre Office
- IntelliJ IDEA, Visual Code ou Visual Studio 2022
- JS, node.js
- VirtualBox ou réel
- PostGréSQL ou Mysql

Pré-requis

- Cours de base de données et POO

Votre mission

Construire le site à partir des spécifications données

Informations utiles

1. **Nom du projet**

   Aventures alpines

2. Configuration de base

Commencez par mettre en place l'environnement de développement avec Node.js, npm (ou yarn), et Create React App sous Windows :

a) **Installer Node.js :** Téléchargez et installez la version de Node.js pour Windows depuis le site officiel de Node.js (https://nodejs.org/)
b) **Installez Create React App :** Ouvre une invite de commande (CMD) ou PowerShell et exécute la commande suivante pour installer Create React App de manière globale :

```
npm install -g create-react-app
```

c) **Créez un nouveau projet :** Une fois Create React App installé, créez un nouveau projet en ouvrant une invite de commande et en exécutant :

```
npx create-react-app sport-montagne-app
```

d) **Accédez au répertoire du projet :** Utilisez la commande `cd` pour te déplacer dans le répertoire du projet :

```
cd sport-montagne-app
```

e) **Lancez le serveur de développement :** Utilisez la commande suivante pour démarrer le serveur de développement :

```
npm start
```

Après avoir exécuté npm start, votre application React sera en cours d'exécution localement sur un serveur de développement. Par défaut, l'application sera accessible via un navigateur web à l'adresse **http://localhost:3000.**

Pour commencer à développer et voir votre application en action, suivez ces étapes :

• Ouvrez un navigateur web (tel que Google Chrome, Mozilla Firefox, etc.).

• Dans la barre d'adresse du navigateur, saisissez l'URL suivante et appuyez sur Entrée **http://localhost:3000**

• Vous devriez maintenant voir votre application React en cours d'exécution dans le navigateur. Vous verrez probablement la page par défaut de React avec le logo React.

• Pour commencer à développer, ouvrez votre éditeur de code (par exemple, Visual Studio Code) et accédez au répertoire de votre projet.

• Modifier **les composants** : Dans votre éditeur de code, ouvrez les fichiers de composants React que vous souhaitez modifier (par exemple, `Activites.js`, `AjouterActivite.js`, etc.). Vous pouvez commencer à éditer ces fichiers pour créer ou modifier des fonctionnalités.

• Afficher **les modifications en temps réel** : Une fois que vous avez apporté des modifications à votre code, enregistrez les fichiers. L'application React se rechargera automatiquement dans le navigateur, vous permettant de voir instantanément les modifications que vous avez apportées.

• Consulter **la console du navigateur** : Si vous rencontrez des erreurs ou des problèmes, ouvrez la console du navigateur en appuyant sur la touche `F12` ou en cliquant avec le bouton droit de la souris puis en sélectionnant "Inspecter" (dans Google Chrome, par exemple). La console affiche les messages d'erreur et les avertissements qui peuvent vous aider à résoudre les problèmes.

• Utiliser **les outils de développement** : Les navigateurs offrent des outils de développement puissants qui vous permettent d'inspecter les éléments de votre page, de surveiller le réseau pour les requêtes API, de déboguer le code JavaScript, etc.

• Sauvegarder **régulièrement** : N'oubliez pas de sauvegarder régulièrement vos fichiers pour éviter de perdre des modifications importantes.

3. Structure du projet

```
java
react-sport-montagne/
  client/                 // Dossier client pour l'application React
    public/
      index.html          // Fichier HTML principal
    src/
      components/         // Composants React
        Activites.js      // Composant pour afficher les activités
        AjouterActivite.js // Composant pour ajouter une activité
        ...
      App.js              // Point d'entrée de l'application React
    package.json
  server/                 // Dossier serveur pour l'API Node.js
    node_modules/
    server.js             // Serveur Express
    package.json
  package.json            // Package.json principal pour le projet
global
```

```
sport-montagne-app/
|-- src/
|   |-- components/
|   |   |-- Header.js
|   |   |-- Footer.js
|   |   |-- Article.js
|   |   |-- Video.js
|   |   |-- HikingRoute.js
|   |-- pages/
|   |   |-- Home.js
|   |   |-- Sports.js
|   |   |-- Articles.js
|   |   |-- Videos.js
|   |   |-- Routes.js
|   |-- data/
|   |   |-- articles.js
|   |   |-- videos.js
|   |   |-- routes.js
|   |-- App.js
|   |-- index.js
|-- public/
|   |-- index.html
|   |-- ...
|-- package.json
|-- ...
```

4. **Contenu du projet**

- **Header.js** : La barre de navigation pour accéder aux différentes sections du site.
- **Footer.js** : Le pied de page avec des informations supplémentaires.
- **Article.js** : Un composant pour afficher les articles sur les sports de montagne.
- **Video.js** : Un composant pour afficher des vidéos liées aux sports de montagne.
- **HikingRoute.js** : Un composant pour afficher des informations sur des itinéraires de randonnée en montagne.

Dans le dossier `data`, mettre des fichiers JSON comme `articles.js`, `videos.js`, et `routes.js` pour stocker les données des articles, vidéos et itinéraires.

5. **Pages**

- **Home.js** : La page d'accueil avec une introduction au site et des liens vers d'autres sections.
- **Sports.js** : Une page qui liste les différents sports de montagne et permet d'en savoir plus sur chacun d'eux.
- **Articles.js** : Une page qui affiche une liste d'articles sur les sports de montagne en utilisant le composant Article.

- **Videos.js** : Une page qui affiche une liste de vidéos en utilisant le composant Video.
- **Routes.js** : Une page qui affiche des itinéraires de randonnée en utilisant le composant HikingRoute.

6. Données

Dans les fichiers JSON tels que `articles.js`, voici un exemple de structure :

```javascript
// articles.js
const articles = [
  {
    id: 1,
    title: "Les bases de l'alpinisme",
    content: "..."
  },
  // Autres articles
];

export default articles;
```

7. Routes

Définissez les routes de votre application en utilisant une librairie de routage comme `react-router-dom`.

8. Styles

Utilisez des bibliothèques de styles comme `styled-components` ou `SASS` pour styliser l'application et lui donner une apparence attrayante.

9. Interactivité

Ajoutez des fonctionnalités interactives comme la possibilité de trier les articles, de filtrer les vidéos par sport, ou de rechercher des itinéraires de randonnée.

10. Partage d'expérience

Ajoutez une section où les utilisateurs peuvent soumettre leurs propres expériences en montagne, avec des photos et des récits.

Mission 3 Configurer la BDD aventures alpines

Compétences	• C1.1.1.2 Identifier les fonctionnalités attendues du service à produire • C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact • C1.1.2.1 Analyser les interactions entre services
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Créer** la BDD du projet Aventures Alpines

Objectifs intermédiaires

- **Réinvestir** les notions de cours en base de données et POO

Outils nécessaires

- Suite Microsoft ou Libre Office
- IntelliJ IDEA, Visual Code ou Visual Studio 2022
- JS, node.js
- VirtualBox ou réel
- PostGréSQL ou Mysql

Pré-requis

- Cours de base de données et POO

Votre mission

Créer et **configurer** la BDD aventures alpines

Informations utiles

L'ajout d'une base de données au projet "Aventures alpines" permettra de stocker et gérer les informations sur les activités, les utilisateurs, les articles de blog, etc.

Structure la base de données

Tables de la base de données :

1. **Activités :**
   - id (Clé primaire)
   - nom
   - description
   - niveau_difficulte
   - image_url
   - saison_recommandee
2. **Sites_Escalade :**
   - id (Clé primaire)
   - nom
   - description
   - niveau_difficulte
   - emplacement
   - image_url
3. **Stations_Ski :**
   - id (Clé primaire)
   - nom
   - description
   - conditions_enneigement
   - emplacement
   - image_url
4. **Utilisateurs :**
   - id (Clé primaire)
   - nom_utilisateur
   - email
   - mot_de_passe (haché et salé)
   - date_inscription
5. **Articles_Blog :**
   - id (Clé primaire)
   - titre
   - contenu
   - date_publication
   - auteur_id (Clé étrangère vers la table Utilisateurs)

Attention : des clefs étrangères sont manquantes (exprès) dans cette description. Toutes vos tables doivent être liées si vous voulez pouvoir manipuler votre BDD.

Intégration de la base de données avec le projet React

1. Utilisez un système de gestion de bases de données (par exemple, MySQL, PostgreSQL) pour créer et gérer les tables mentionnées ci-dessus.
2. Utilisez une API côté serveur pour interagir avec la base de données. Utilisez la technologie Node.js et Express.
3. Implémentez des points d'extrémité API pour effectuer des opérations CRUD (Création, Lecture, Mise à jour, Suppression) sur les données de la base de données.
4. Dans le projet React, utilisez des requêtes HTTP (par exemple, avec la bibliothèque Axios) pour appeler les points d'extrémité API et récupérer les données.
5. Mettez en place un état global dans votre application React à l'aide de contextes (par exemple, React Context) pour stocker temporairement les données récupérées de la base de données et les rendre disponibles à différents composants.
6. Affichez les données de la base de données dans les composants appropriés de votre application React, par exemple, affichez les activités sur la page "Activités", les articles de blog sur la page "Blog", etc.

Gérer les opérations de lecture (GET)

Gérer l'intégration de la base de données avec le projet React en utilisant une API côté serveur pour gérer les opérations de lecture (GET). Dans cet exemple, nous allons supposer que vous utilisez Node.js et Express pour créer l'API côté serveur et Axios pour effectuer des requêtes HTTP depuis React.

Étape 1 : Mise en place de l'API côté serveur

a) Installez Node.js et npm (Node Package Manager) sur votre serveur.
b) Créez un nouveau dossier pour votre API et initialisez un projet Node.js :

```
mkdir react-sport-montagne-api
cd react-sport-montagne-api
npm init -y
```

c) Installez les dépendances nécessaires (Express, CORS, etc.) :

```
npm install express cors
```

d) Créez un fichier server.js dans le dossier de l'API et configurez-le pour mettre en place un serveur Express :

```
const express = require('express');
const cors = require('cors');
const app = express();
const PORT = process.env.PORT || 5000;

app.use(cors());

// Routes pour les opérations de lecture
app.get('/api/activites', (req, res) => {
    // Ici, vous devriez utiliser une méthode pour récupérer les données de
    const activites = [
        // Données de la base de données
    ];
    res.json(activites);
});

// D'autres routes pour d'autres opérations de lecture

app.listen(PORT, () => {
    console.log(`Serveur en cours d'écoute sur le port ${PORT}`);
});
```

Étape 2 : Effectuer des requêtes depuis React

a) Dans votre projet React, utilisez Axios pour effectuer des requêtes HTTP depuis le côté client. Installez Axios dans votre projet :

```
npm install axios
```

b) Dans le composant React où vous souhaitez afficher les données (par exemple, la page "Activités"), importez Axios et utilisez-le pour récupérer les données depuis l'API :

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

const ActivitesPage = () => {
    const [activites, setActivites] = useState([]);

    useEffect(() => {
        axios.get('/api/activites')
            .then(response => {
                setActivites(response.data);
            })
            .catch(error => {
                console.error('Erreur lors de la récupération des activités
            });
    }, []);

    // Utilisez la variable 'activites' pour afficher les données dans votr

    return (
        // Votre JSX pour afficher les données
    );
};

export default ActivitesPage;
```

Cette structure vous permet de récupérer des données depuis la base de données à l'aide de votre API et de les afficher dans votre application React. Vous pouvez étendre ce modèle pour inclure d'autres opérations CRUD et pour gérer les états, les erreurs, la pagination, etc.

N'oubliez pas d'adapter les noms de routes, les méthodes et les structures de données en fonction des besoins.

Gérer les opérations d'écriture (POST)

Gérer les opérations d'écriture (POST) dans votre API côté serveur et dans votre application React en utilisant Axios pour envoyer des données à la base de données.

Étape 1 : Côté Serveur - Gestion des opérations POST

1. Dans votre fichier server.js, ajoutez une route pour gérer les opérations de création (POST) :

```
app.use(express.json()); // Middleware pour parser le JSON

app.post('/api/activites', (req, res) => {
    // Ici, vous devriez utiliser une méthode pour insérer les données dans
    const nouvelleActivite = req.body; // Les données du formulaire POST
    // Insérez la nouvelleActivite dans la base de données
    res.status(201).json({ message: 'Activité créée avec succès' });
});

// D'autres routes pour d'autres opérations POST
```

Étape 2 : Côté Client - Gestion des opérations POST dans React

1. Dans le composant React où vous avez un formulaire pour ajouter une nouvelle activité (par exemple, une page "Ajouter une Activité"), utilisez Axios pour envoyer les données à l'API :

Étape 2 : Configuration de l'API côté serveur

1. Dans le dossier `server`, créez un fichier `server.js` pour configurer votre serveur Express.

```
javascript
const express = require('express');
const cors = require('cors');
const bodyParser = require('body-parser');
const app = express();
const PORT = process.env.PORT || 5000;

app.use(cors());
app.use(bodyParser.json());

// Routes pour les opérations de lecture et d'écriture
// Exemple : require('./routes/activites')(app);

app.listen(PORT, () => {
    console.log(`Serveur en cours d'écoute sur le port ${PORT}`);
});
```

2. Créez un dossier `routes` pour stocker vos fichiers de routage API (par exemple, `activites.js`) et définissez les routes nécessaires pour effectuer des opérations CRUD.

Mission 4 Configurer l'application aventures alpines

Compétences	• C1.1.1.2 Identifier les fonctionnalités attendues du service à produire • C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact • C1.1.2.1 Analyser les interactions entre services
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Créer** l'application du projet Aventures Alpines

Objectifs intermédiaires

- **Réinvestir** les notions de cours en base de données et POO

Outils nécessaires

- Suite Microsoft ou Libre Office
- IntelliJ IDEA, Visual Code ou Visual Studio 2022
- JS, node.js
- VirtualBox ou réel
- PostGréSQL ou Mysql

Pré-requis

- Cours de base de données et POO

Votre mission

Créer et **configurer** l'application aventures alpines sous REACT à l'aide des informations utiles

Informations utiles

1. Dans le dossier `client`, initialisez une application React.

```
bash
cd client
npx create-react-app .
```

2. Créez des composants React pour chaque section de votre site (par exemple, `Activites.js`, `AjouterActivite.js`, `Blog.js`, etc.).

Exemple de l'écriture du composant Activites.js

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
const Activites = () => {
  const [activites, setActivites] = useState([]);
  const [loading, setLoading] = useState(true);
  useEffect(() => {
    // Effectuer une requête GET vers votre API pour récupérer la liste des activités
    axios.get('/api/activites')
      .then(response => {
        setActivites(response.data);
        setLoading(false);
      })
      .catch(error => {
        console.error('Erreur lors de la récupération des activités :', error);
        setLoading(false);
      });
  }, []);
  return (
    <div className="activites">
```

```
        <h2>Liste des Activités</h2>
        {loading ? (
          <p>Chargement en cours...</p>
        ) : (
          <ul>
            {activites.map(activite => (
              <li key={activite.id}>
                <h3>{activite.nom}</h3>
                <p>{activite.description}</p>
                <p>Niveau de difficulté : {activite.niveau_difficulte}</p>
                {/* Ajoutez d'autres détails de l'activité ici */}
              </li>
            ))}
          </ul>
        )}
      </div>
  );
};  export default Activites;
```

Utilisez Axios pour effectuer des requêtes HTTP vers votre API côté serveur à partir de vos composants React (voir **Étape 2 : Effectuer des requêtes depuis React)**

Mission 5 Créer la maquette du site web aventures alpines

Compétences	• C1.1.1.2 Identifier les fonctionnalités attendues du service à produire • C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact • C1.1.2.1 Analyser les interactions entre services
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Créer** la maquette du site web de l'application du projet Aventures Alpines

Objectifs intermédiaires

- **Réinvestir** les notions de cours en base de données et POO

Outils nécessaires

- Suite Microsoft ou Libre Office
- IntelliJ IDEA, Visual Code ou Visual Studio 2022
- JS, node.js
- VirtualBox ou réel
- PostGréSQL ou Mysql

Pré-requis

- Cours de base de données et POO

Votre mission

Créer et **configurer** la maquette de l'application aventures alpines sous REACT à l'aide des informations utiles

Informations utiles

Page d'accueil

- En-tête avec le logo du projet et une navigation claire vers les différentes sections du site.
- Bannière attrayante mettant en avant des images inspirantes de sports de montagne.
- Section "Activités Populaires" affichant des vignettes cliquables menant aux pages dédiées à chaque activité.

Page Activités

- Présentation des activités sportives en montagne, telles que la randonnée, l'escalade, le ski, etc.
- Carrousels d'images pour chaque activité, montrant des photos captivantes.
- Boutons "En savoir plus" redirigeant vers des pages détaillées pour chaque activité.

Page Randonnée

- Informations spécifiques sur la randonnée en montagne : niveaux de difficulté, équipement recommandé, meilleures saisons, etc.
- Carte interactive affichant les sentiers de randonnée populaires avec des points d'intérêt.
- Galerie de photos de randonnées passées.

Page Escalade

- Introduction à l'escalade en montagne avec des conseils de sécurité et des astuces pour les débutants.
- Liste des sites d'escalade renommés, classés par niveau de difficulté.
- Intégration de vidéos montrant des grimpeurs expérimentés en action.

Page Ski

- Présentation des différentes disciplines de ski en montagne : ski alpin, ski de fond, ski hors-piste, etc.
- Conditions d'enneigement en temps réel pour les stations de ski partenaires.
- Témoignages de skieurs passionnés et liens vers des offres spéciales.

Page Blog

- Articles et récits d'aventures en montagne rédigés par des experts et des passionnés.
- Possibilité de trier les articles par catégorie, popularité ou date.
- Formulaire d'abonnement à la newsletter pour rester informé des dernières nouvelles.

Page Contact

- Formulaire de contact pour poser des questions ou demander des informations supplémentaires.

- Coordonnées de contact, y compris une carte montrant l'emplacement de l'entreprise ou des points d'intérêt en montagne.
- Liens vers les réseaux sociaux pour rester connecté

```
import React, { useState } from 'react';
import axios from 'axios';

const AjouterActivitePage = () => {
    const [nouvelleActivite, setNouvelleActivite] = useState({
        nom: '',
        description: '',
        niveau_difficulte: '',
        // Autres champs
    });

    const handleInputChange = event => {
        const { name, value } = event.target;
        setNouvelleActivite(prevState => ({
            ...prevState,
            [name]: value
        }));
    };

    const handleSubmit = event => {
        event.preventDefault();
        axios.post('/api/activites', nouvelleActivite)
            .then(response => {
                console.log(response.data.message);
                // Réinitialisez le formulaire ou effectuez une autre actio
            })
            .catch(error => {
                console.error('Erreur lors de la création de l\'activité :'
            });
    };
```

```
    return (
        // Votre formulaire JSX avec des champs pour les données de la nouv
        // Assurez-vous d'ajouter des gestionnaires d'événements pour handl
    );
};
```

Remarques

- Assurez-vous d'ajouter des champs de formulaire correspondant aux données que vous attendez pour une nouvelle activité dans le composant `AjouterActivitePage`.
- Le middleware `express.json()` est utilisé pour analyser les données JSON envoyées depuis le client.
- Dans le composant React, utilisez les gestionnaires d'événements `handleInputChange` et `handleSubmit` pour gérer la saisie de l'utilisateur et l'envoi des données au serveur.
- Assurez-vous de gérer les erreurs correctement en cas d'échec de l'opération POST

Mission 6 Gérer la planification des tâches avec Trello

Compétences	• C1.1.1.2 Identifier les fonctionnalités attendues du service à produire • C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact • C1.1.2.1 Analyser les interactions entre services
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Utiliser** le logiciel TRELLO pour gérer la planification des tâches

Objectifs intermédiaires

- **Réinvestir** les notions de cours en gestion de projets

Outils nécessaires

- Suite Microsoft ou Libre Office
- IntelliJ IDEA, Visual Code ou Visual Studio 2022
- JS, node.js
- VirtualBox ou réel
- PostGréSQL ou Mysql
- TRELLO

Votre mission

Créer à l'aide du logiciel Trello, la planification des tâches du projet

Mission 7 Gérer les tests de développement par flux

Compétences	• C1.1.1.2 Identifier les fonctionnalités attendues du service à produire • C1.1.1.1 Recenser et caractériser les contextes d'utilisation, les processus et les acteurs sur lesquels le service à produire aura un impact • C1.1.2.1 Analyser les interactions entre services
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Utiliser** le logiciel SELENIUM pour gérer la planification des tâches

Objectifs intermédiaires

- **Réinvestir** les notions de cours en gestion de projets

Outils nécessaires

- Suite Microsoft ou Libre Office
- IntelliJ IDEA, Visual Code ou Visual Studio 2022
- JS, node.js
- VirtualBox ou réel
- PostGréSQL ou Mysql
- SELENIUM

Votre mission

Créer à l'aide du logiciel SELENIUM, les tests de développement par flux

Documenter l'application

Mission 1 Documenter l'application Aventures alpines

Compétences	A4.1.5 Prototypage de composants logiciels — Non — C4.1.5.1 Choisir les éléments de la solution à prototyper; C4.1.5.2 Développer un prototype; C4.1.5.3 Valider un prototype A4.1.6 Gestion d'environnements de développement et de test — Non — C4.1.6.1 Mettre en place et exploiter un environnement de développement; C4.1.6.2 Mettre en place et exploiter un environnement de test A4.1.7 Développement, utilisation ou adaptation de composants logiciels — Non — C4.1.7.1 Développer les éléments d'une solution; C4.1.7.2 Créer un composant logiciel; C4.1.7.3 Analyser et modifier le code d'un composant logiciel; C4.1.7.4 Utiliser des composants d'accès aux données; C4.1.7.5 Mettre en place des éléments de sécurité liés à l'utilisation d'un composant logiciel
Vocabulaire à connaître	
Evaluation	Production d'écrit DST E4

Objectif principal

- **Documenter** l'application Aventures alpines

Objectifs intermédiaires

- **Réinvestir** les notions de cours de java
- **Créer** un document technique de l'application aventures alpines

Outils nécessaires

- Suite Microsoft ou Libre Office

Pré-requis

- Cours de base de données
- Cours sur Java

Votre mission

Créer une fiche de synthèse de l'application Aventures alpines selon les indications ci-dessous

Sommaire ou table des matières

1. **Automatiser** votre sommaire

Introduction

1. **Présenter**
   1.1. le contexte sur lequel vous travaillez (Description d'aventures alpines)

Déroulé de la mission

1. **Donner** votre schéma UML associé au contexte
2. **Préciser** les contraintes du cahier des charges
1- **Présenter** Node JS
   a. Composition
   b. version
   c. Paramétrage avec copie écran
   d. Emplacement physique sur le serveur (chemin, dossier)
2- **Montrer** à l'aide de copie écran la structure de votre application
3- **Expliquer** les fonctionnalités de votre programme

Analyse de l'activité

1. **Spécifier** les difficultés rencontrées au cours des différentes phases de mise en place
2. **Lister** les avantages de sa mise en place pour une entreprise
3. **Mentionner** les apports professionnels acquis à travers cette expérience
4. **Préciser** les apports personnels acquis à travers cette expérience