



# SportSee

Construire un tableau de Bord analytique avec React

Développeur chez sportsee, une startup de coaching en pleine croissance.

L'entreprise lance une nouvelle version de la page "profil" d'un utilisateur. Elle permettra à l'utilisateur de suivre le nombre de séances effectuées ainsi que le nombre de calories brûlées

# OBJECTIF CAHIER DES CHARGES

- Refaire la page Profil avec React
- Intégration des graphiques
- Responsive 1024 x 780 pixels
- Récupérer les données du backend
- Réaliser le mock des données
- Utiliser node Js
- Intégrer l'api
- Réaliser le mock des données
- Documentation en anglais
- Intégrer des PropTypes pour chacun des composants



Le projet utilise **React**, bibliothèque **JavaScript** pour la construction d'interfaces utilisateur interactives.

Les dépendances du projet incluent **Axios** pour effectuer des requêtes HTTP, **Recharts** pour la visualisation de données et **npm** pour gérer les dépendances et les scripts de construction et de démarrage.

**Node.js** doit également être installé. Les dépendances peuvent être installées en utilisant la commande **npm install** dans le répertoire racine du projet.

---

Node.js version v18.14.0

---

npm version 8.4.1

---

Visual studio code version 1.75.1

---

React version 18.2.0

---

Recharts version 2.4.2

---

Sass version 1.58.1

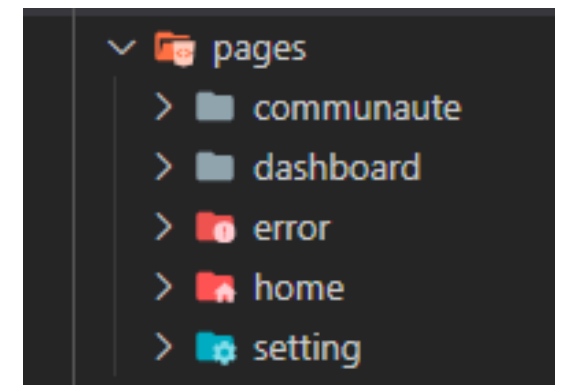
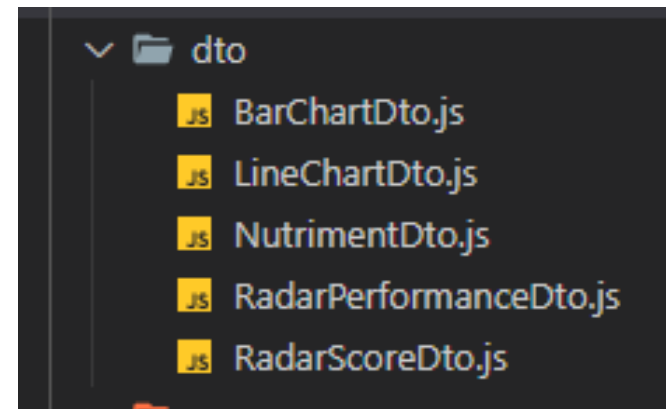
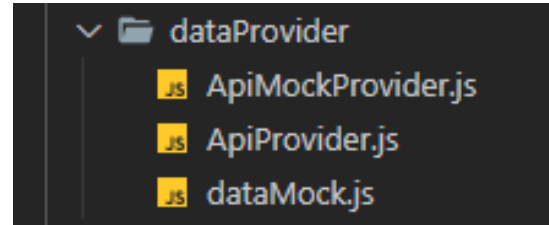
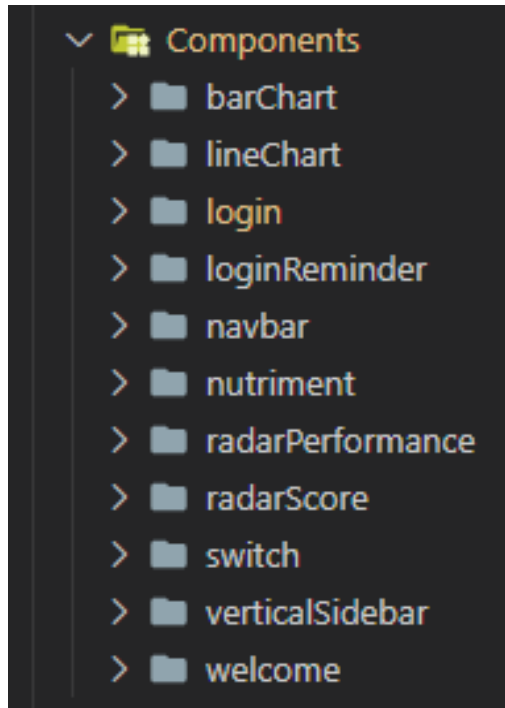
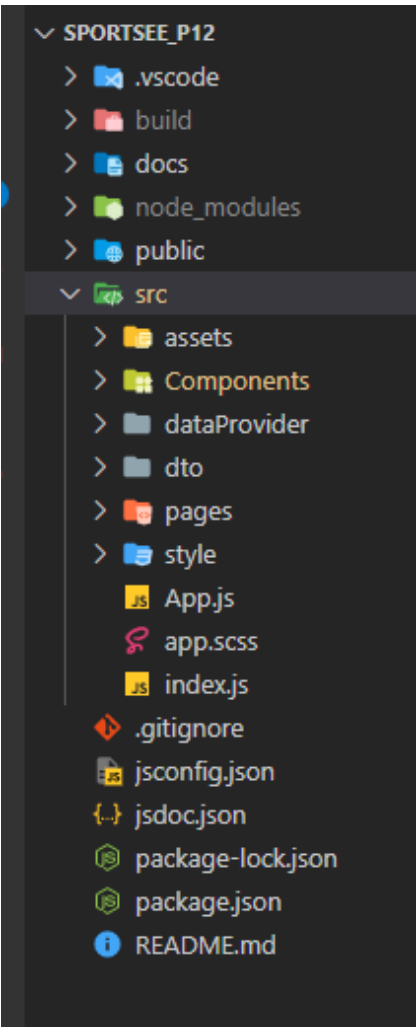
---

Axios version 1.3.3

---



# ARCHITECTURE du Projet



```
const App = () => {  
  return (  
    <header className="routePage">  
      <Navbar />  
      <VerticalSidebar />  
      <section className="content">  
        <Routes>  
          <Route path="/" element={<Home />} />  
          <Route path="/dashboard/:userId/:isDemo" element={<Dashboard />} />  
          <Route path="/setting/" element={<Setting />} />  
          <Route path="/communaute/" element={<Communaute />} />  
          <Route path="*" element={<Error />} />  
          <Route path="/login-reminder" element={<LoginReminder />} />  
        </Routes>  
      </section>  
    </header>  
  );  
};
```

# Routes - App.js

# ACCUEIL

SportSee

Accueil

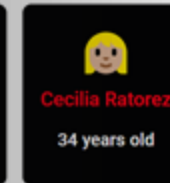
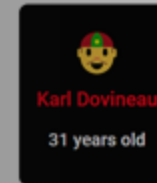
Profil

Réglage

Communauté



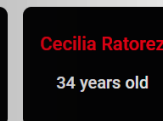
Choose and log in



Datas  
Mockees



Choose and log in



Datas  
API

# LOADER Page Dashboard

Hook



In Progress

```
21  const [isLoading, setIsLoading] = useState(true);
```

```
61  // Simulate loading for 5 seconds
62  await new Promise((resolve) => setTimeout(resolve, 1000));
63  setFirstName(firstName);
```

Crée une promesse asynchrone qui fait attendre pendant une seconde (1000 millisecondes) avant de continuer.

```
70  setIsLoading(false);
71  } catch (error) {
72    setIsLoading(false);
73  }
74  }
```

`setIsLoading(false)` attribue la valeur `false` à la variable `isLoading`  
= indique le chargement des données terminé.

# Class ApiProvider

La classe **ApiProvider** est une classe qui fournit des fonctions pour récupérer des données utilisateur à partir d'une API. Elle utilise la bibliothèque Axios pour effectuer des requêtes HTTP.

La classe comporte plusieurs méthodes pour afficher des graphiques

Elle est conçue pour faciliter l'accès aux données utilisateur à partir d'une API en utilisant des méthodes simples et en retournant les résultats sous forme d'objets DTO spécifiques.

```
class ApiProvider {
  constructor() {
    this.baseUrl = BASE_URL;
  }

  /**
   * Handles errors when retrieving user data.
   * @param {Error} error - The error generated when retrieving the data.
   * @throws {Error} - An error indicating that user data cannot be fetched.
   */
  handleError(error) {
    console.log("Error fetching user data: ", error);
    console.error("Error fetching user data: ", error);
    throw new Error("Unable to fetch user data");
  }

  /**
   * Retrieves the user's first name by user ID.
   * @param {string} userId - The user ID.
   * @returns {Promise<string|null>} - A promise that resolves with the user's first name
   */
  async getUsernameByUserId(userId) {
    // Check if the request URL is correct
    console.log("Request URL: ", this.baseUrl + userId);

    // Make a GET request using axios
    return axios
      .get(this.baseUrl + userId)
      .then((response) => {
        console.log("response: ", response);

        return response.data &&
          response.data.data &&
          response.data.data.userInfos &&
          response.data.data.userInfos.firstName
            ? response.data.data.userInfos.firstName
            : null;
      })
  }
}
```



```

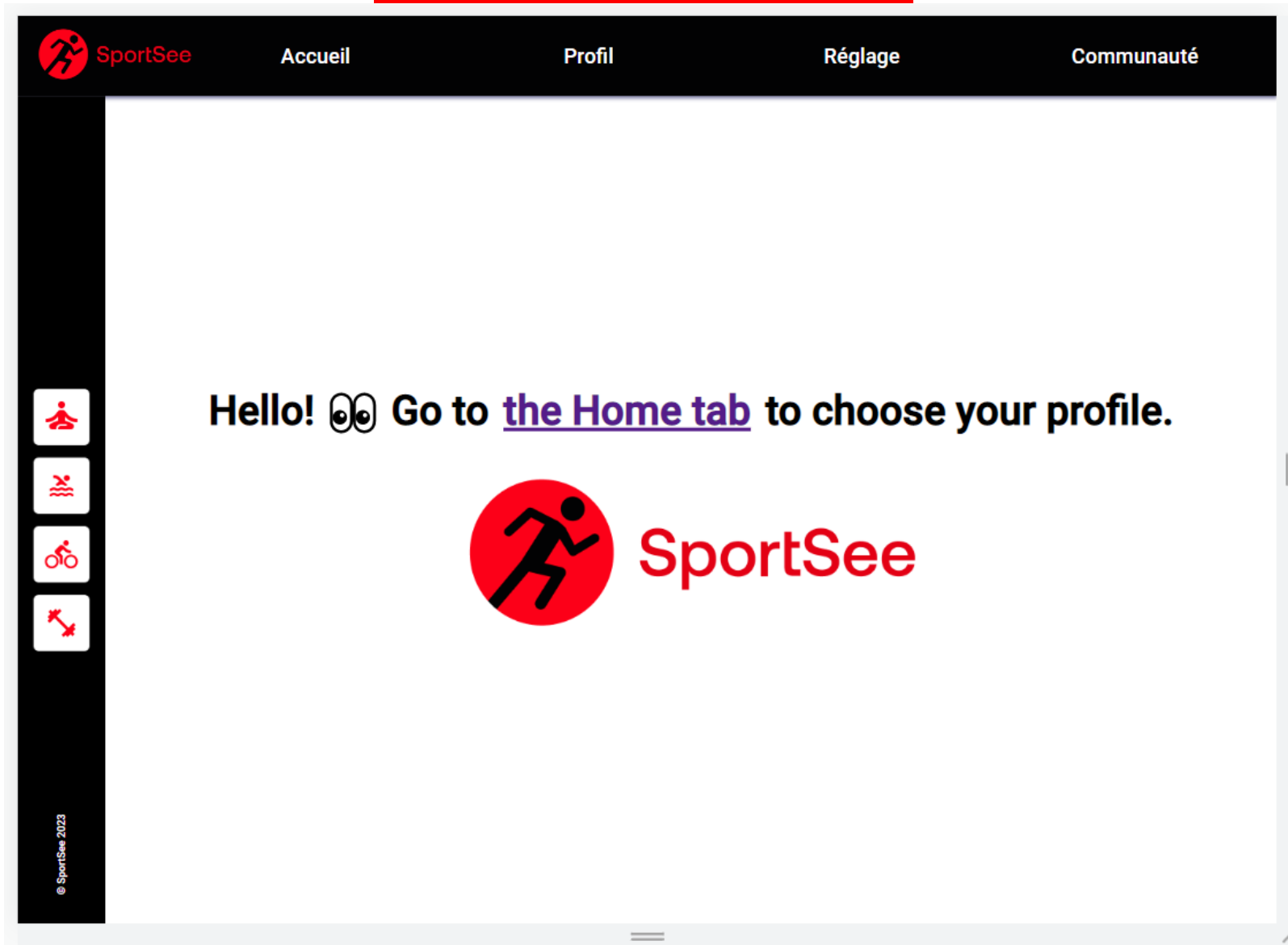
class ApiMockProvider {
  /**
   * Retrieves user activities by user ID.
   * @param {number} userId - The user ID.
   * @returns {BarChartDto} - User activities in the form of BarChartDto
   * @memberof ApiMockProvider
   */
  getActivitiesByUserId(userId) {
    const userSessions = [];
    const currentUser = USER_ACTIVITY.find(
      (user) => user.userId === parseInt(userId)
    );
    if (currentUser) {
      currentUser.sessions.forEach((session) => {
        const day = session.day.split("-");
        const formattedDate = `${day}`;
        userSessions.push({
          day: formattedDate,
          kilogram: session.kilogram,
          calories: session.calories,
        });
      });
    }
    return new BarChartDto(
      userSessions,
      "Day",
      "Kilograms",
      "Calories"
    );
  }
}

```

# Class ApiMockProvider

- La classe **ApiMockProvider** est une classe qui fournit des fonctions pour récupérer des données fictives (mockées) liées à un utilisateur. Elle offre des méthodes pour récupérer différentes informations.
- Aussi conçue pour faciliter la récupération de données fictives pour des tests ou des simulations.

# Profil



Connectez-vous  
pour être redirigé  
vers votre profil

# CONCLUSION

Un tableau de bord d'analyse avec **React** pour la startup **SPORTSEE** dédié au coaching sportif a été développé.

Le projet utilise un **backend** pour les appels HTTP et intègre une **API** pour récupérer les données.

Une classe de modélisation de données a été créée pour garantir que les données de l'**API** sont toujours formatées avant d'être utilisées

La bibliothèque graphique **Récharts** a été utilisée pour la partie graphique.

Le projet respecte plusieurs contraintes techniques telles que l'intégration de **Proptypes** pour **chaque composé React** .

Le résultat final est un **tableau de bord** d'analyse interactif , attrayant dédiée au coaching sportif.