Présentation du projet 12

SportSee

II

Contexte du projet

SportSee est une startup dédiée au coaching sportif. En pleine croissance, l’entreprise souhaite lancer une nouvelle version de la page profil de l’utilisateur (desktop). Cette page va notamment permettre à l’utilisateur de suivre le nombre de sessions réalisées ainsi que le nombre de calories brûlées. Pour ce sprint, j'ai dû intégrer les users stories de la partie To Do figurant sur le Kanban du projet. J'avais également à ma disposition une maquette sur Figma et le dossier backend, que j'ai cloné.

Outils utilisés pour l'implémentation

- Bibliothèques pour l'interface utilisateur

  React et React Router
  Styled components
  Recharts

- Documentation technique du projet

  JS Doc
  Proptypes

IV

Structure de l'application et récupération des données

V

Le routeur

- Contient la structure des pages réalisées (avec le layout).

src > router > routes.jsx > router

```jsx
import {
  createBrowserRouter,
  createRoutesFromElements,
  Route,
} from "react-router-dom";
import Layout from "../components/Layout";
import Home from "../pages/Home";
import Dashboard from "../pages/Dashboard";
import NotFound from "../pages/NotFound";

export const router = createBrowserRouter(
  createRoutesFromElements(
    <Route path="/" element={<Layout />}>
      <Route index element={<Home />} />
      <Route path="user/:userId" element={<Dashboard />} />
      <Route path="*" element={<NotFound />} />
      <Route path="404" element={<NotFound />} />
    </Route>
  )
);
```

src > index.jsx

```jsx
import React from "react";
import ReactDOM from "react-dom/client";
import { RouterProvider } from "react-router-dom";
import { GlobalStyle } from "./style/GlobalStyle";
import { router } from "./router/routes.jsx";

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <RouterProvider router={router} />
    <GlobalStyle />
  </React.StrictMode>
);
```

- Est appelé dans index.jsx qui contient également le GlobalStyle de l'application.

Le mock des données et les calls API

Les calls sont réalisés dans le fichier api.jsx

Création d'une variable *callApi*. Si sa valeur est *true*, cela permettra d'appeler l'API et si sa valeur est *false*, cela permettra d'appeler les données mockées. Les 4 fonctions de call exécutent l'une ou l'autre des instructions conditionnelles selon la valeur de *callApi*.

Données mockées

Données copiées dans un fichier JS
Import sous forme de modules
Récupération dans ma fonction fetch dans un objet "*data*" pour avoir la même structure que les données de l'API

Calls API

Fetch des 4 endpoints donnés dans la documentation de l'API

Le fichier api

L'id de l'utilisateur est passé en paramètre pour réaliser le fetch des données

```
import {
  USER_MAIN_DATA,
  USER_ACTIVITY,
  USER_PERFORMANCE,
  USER_AVERAGE_SESSIONS,
} from "./mocked-data/data";

let callApi = false;

/**
 * Call API to retrieve user main data, activities, performances and average sessions
 * @param { String } userId
 * @return { Object }
 */

export async function fetchUserMainData(userId) {
  if (callApi) {
    const response = await fetch(`http://localhost:3000/user/${userId}`);
    let data = await response.json();
    return data;
  } else {
    return { data: USER_MAIN_DATA.find((element) => element.id == userId) };
  }
}

export async function fetchUserActivity(userId) {
  if (callApi) {
    const response = await fetch(
      `http://localhost:3000/user/${userId}/activity`
    );
    let data = await response.json();
    return data;
  } else {
    return { data: USER_ACTIVITY.find((element) => element.userId == userId) };
  }
}
```

VIII

Les composants

Les composants parents

- Composants pages

Home
Dashboard
Error

Home permet de choisir l'utilisateur dont on souhaite voir les données.

Error permet une redirection en cas d'erreur dans l'URL ou sur l'id de profil

Dashboard

- contient le state de l'application
- récupère l'id passée en paramètre
- utilise le hook useEffect() et récupère les données issues des fetch du fichier api.jsx
- passe les props à ses enfants

Les composants enfants

- Composants avec affichage dynamique

Profile
FigureCard
Un composant par graphique

Les données sont passées depuis la page User via les props (y compris dans les graphiques).
La mise en forme des graphiques peut passer par la création de fonctions.

Les composants enfants

- Composants statiques

Layout
Header
Sidebar

Layout contenant les 2 autres composants et intégré dans le routeur pour englober toute l'application

Difficultés rencontrées

- Récupération des résultats du fetch dans la page Dashboard

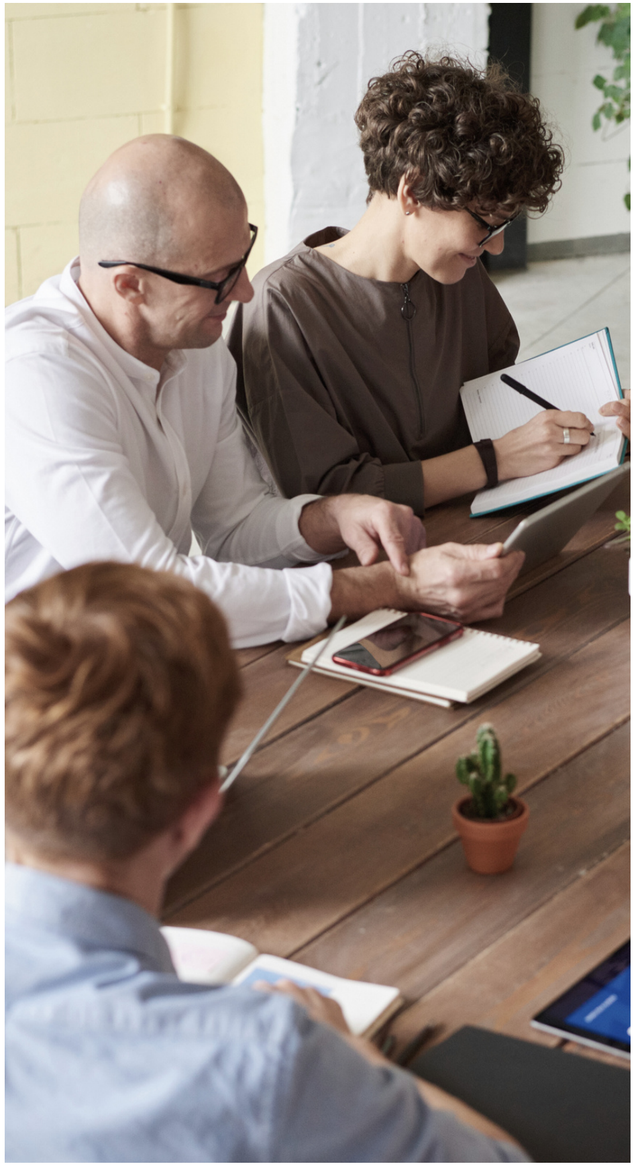
Utilisation du state

- Formatage des données mockées pour correspondre à celles de l'API

Création de l'objet data pour englober les données mockées

- Mise en forme de certains graphiques

Fonctions pour créer des composants personnalisés



Merci !

SportSee