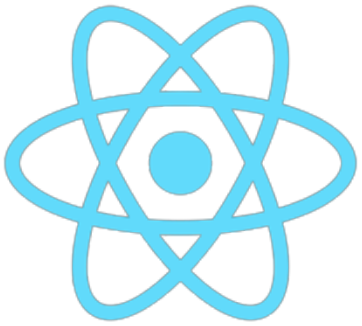
Rapport
Programmable Web - Client side

Année 2021-2022

Rizzo Loïc

GitHub : Loic-Rizzo

Mail : loic.rizzo@etu.univ-cotedazur.fr



Enseignants : *François Michaudon, Hugo Mallet*

POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

Université Nice Sophia Antipolis

Stratégie employée pour la gestion des versions avec Git

Afin de gérer efficacement les versions de notre application avec Git, nous avons adopté une stratégie de *branching* bien connue similaire au Git flow. Notre application dispose d'une branche *main* prête à la production, ainsi qu'une branche *develop* implémentant les dernières fonctionnalités dans un état stable. Chaque fonctionnalité est implémentée en amont sur une branche spécifique à cette fonctionnalité nommée "feature/<nom de la fonctionnalité>".

Toute nouvelle implémentation est définie puis suivie à l'aide des issues Github. Elles nous permettent d'identifier les commits associés à une fonctionnalité.

Tâches effectuées

Thème sombre (3 h)

La possibilité pour l'utilisateur de changer le thème de l'application est complètement implémentée. En effet, les thèmes sombre et clair peuvent être changés à tout moment, tout le site appliquera les changements du mode, allant du background et la police d'écriture, jusqu'au popup Leaflet. Le thème par défaut est celui de préférence choisi par l'utilisateur sur son navigateur. Si l'utilisateur décide de changer de thème, il est sauvegardé pour sa prochaine visite sur le site.

Solutions choisies

Le thème sombre a été implémenté avec la librairie Mantine, en effet Mantine dispose de composants et hooks permettant d'intégrer un thème sombre facilement à l'application. Cette librairie étant déjà présente au sein du projet avant le début du développement du thème sombre, son utilisation est tout à fait naturelle. Il propose de sauvegarder le thème, ajuster selon le thème par défaut du navigateur et même un raccourci clavier pour changer de mode. Il répond donc parfaitement aux contraintes données pour l'implémentation de cette fonctionnalité.

Difficultés rencontrées

Appliquer le thème sombre sur certains composants spécifiques à représenté la principale difficulté. Par exemple, il est possible d'ouvrir un popup pour obtenir des informations à propos d'une station sur la carte en cliquant sur le marqueur correspondant.

Malheureusement le composant Popup de la librairie React Leaflet pour afficher la map n'était pas impacté par le changement de background.

Impacts

Le thème sombre afficher alors des Popups "clair" alors que le mode sélectionné était "Sombre".

Solutions / Blocages

Afin de pallier cette défaillance, il a fallu ajouter un fichier *scss* pour le composant **Map** dans le but de modifier le style du Popup, la première approche a été de changer le *className* du Popup afin de définir son style en fonction du mode. Les Popups Rect Leaflet étant particulier le className n'était pas actualisé lors du render, pour contourner ce problème, j'ai encapsulé le contenu du Popup dans une *div* et modifié selon le mode le style de cette *div* à l'aide du *className*.

Composant optimal

DarkThemeButton.js est le composant React par excellence, petit et précis sur ce qu'il fait, il pourrait être réutilisé sans problème dans d'autres projets pour accomplir une mission similaire.

Mode de visualisation graphique (6 h)

Le mode de visualisation graphique permet de comparer le prix des différents carburants sélectionnés entre les villes choisies à travers un diagramme à barres.

Solutions choisies

Pour afficher un diagramme de qualité j'ai utilisé la librairie react-chartjs-2 sur couche de la libraire Chart.js pour l'utilisation sous forme de composant React.

Difficultés rencontrées

L'affichage du graphique du graphique est très simple grâce à la librairie react-chartjs-2. La difficulté se trouve plutôt avec la communication avec le back, en effet les données présentent sur https://www.prix-carburants.gouv.fr sont très peu formatées pour le nom des villes. Une même ville peut avoir plusieurs orthographes, selon la présence ou non de tirets, accents et majuscules.

Impacts

Récupérer une liste des villes depuis le back n'est donc pas possible en l'état, car plusieurs doublons avec de légères différences orthographiques apparaissent. Même avec un traitement simplet sur les majuscules et accents, les villes avec des espaces et des tirets continuent de poser un problème. On ne peut en conséquence pas proposer à l'utilisateur un choix sur toutes les villes présentent dans la base de données pour construire son graphe.

Solutions / Blocages

Actuellement la solution employée, et l'utilisation d'une constante répertoriant les villes les plus connues et habitées pour laisser un large choix à l'utilisateur afin de construire son diagramme. Dans le futur la solution la plus optimale serait de continuer la normalisation sur les données dans le back pour que chaque ville ait une orthographe unique, de cette manière la partie front-end pourra récupérer la liste des villes sans se soucier du reste.

Composant à revoir

StationChart.js est un composant qui pourrait être amélioré de deux façons. Premièrement, je pense qu'il pourrait être découpé en deux composants, un pour la partie sélection et un autre pour l'affichage du diagramme. Deuxièmement, le code afin de traiter les données pour construire le graphique pourrait être optimisé. Cependant, il a aussi de bonnes idées comme l'utilisation des *useEffects* dans le but d'actualiser le graphique selon les changements de certains paramètres donnés par l'utilisateur.