2021 / 2022

Programmable web
Client side
-
Rapport
-
Hamza Ayoub

Sommaire

1. Identifiant github

Mon identifiant Github est : **hamzaayoub98**

2.Tâches effectués

Concernant les tâches effectuées dans ce projet, on a choisi qu'un binôme travail sur la partie Frontend et l'autre binôme sur la partie Backend. Je faisais partie du binôme qui a travaillé sur le Frontend alors toutes les tâches que j'ai réalisées étaient dans cette partie. Les tâches que j'ai effectuées sont :

- Initier la page Map en affichant la carte d' OpenStreetMap
- Créer la page Table ou on peut afficher les stations dans un tableau : Afficher un tableau qui permet de visualiser les stations avec possibilité de trier les stations selon les l'adresse,ville…
- Créer la page Graph ou on peut visualiser deux graphes qui montrent la moyenne des prix des carburants dans toutes les stations, et un deuxième qui montre l'évolution des prix de ces derniers.
- Etablir la connexion avec le backend
- Parser les données reçues depuis le backend et les adapter afin de pouvoir les utiliser dans les trois pages.
- Adapter l'affichage des stations sur la carte en fonction de la position de l'utilisateur: Afficher que les stations qui sont localisé dans un périmètre prédéfinis autour de l'utilisateur.

3. Stratégie employée pour la gestion de versions Git

On a utilisé Git pour organiser le travail sur notre projet. La branche Main a été utilisée pour push les versions stables des fonctionnalités qu'on ajoute sur notre projet. On a aussi utilisé d'autres branches afin de travailler sur d'autres fonctionnalités séparément jusqu'à leur finalisation afin de ne pas risquer d'avoir des conflits avec les commits des autres membres d'équipe qui travaillent sur d'autres fonctionnalités.

4. Solutions choisies

On a effectué plusieurs choix au cours de ce projet, parmis ces choix on commence par le choix d'utiliser OpenStreetMap comme carte dans notre projet, évidemment il y a plusieurs alternatives comme google maps et d'autres cartes, mais le choix d'utiliser OpenStreetMap a été fait d'abord car c'est gratuit contrairement à Google Maps , en plus la documentation

de la librairie ‘react-leaflet’ est bien détaillée, et on peut trouver beaucoup de réponses proposées par la communauté en ligne pour les différents problèmes qu’on peut rencontrer.

Pour afficher les données dans un tableau j’ai utilisé la librairie “@material-ui/core” qui permet de créer des tableaux totalement personnalisables selon le besoin. Du coup J’ai choisi de créer un tableau avec des lignes qui sont cliquables et qui affiche plus d’informations en dessous de la ligne. J’ai choisi d’afficher les données dans le tableau selon ce modèle afin de permettre à l’utilisateur de filtrer selon les adresses des stations et puis consulter les prix du carburant dans chaque station en cliquant sur la ligne correspondante à la station.

La réception de données se fait dans le fichier “APP.jsx” où les données sont traitées et préparées à être envoyées vers les composants en utilisant les props.

5. Difficultés rencontrées

Lors du développement des différentes tâches, on a rencontré plusieurs difficultés qui bloquent parfois l’avancement. Parmis ces difficultées on trouve le bon affichage des stations sur la map et sur le tableau, le fichier contenait plusieurs cas particuliers qui fallait prendre en considération sinon on recevait des erreurs, parmis ces cas particuliers on trouve par exemple des stations avec des listes de services ou de prix vides.
Un autre problème qu’on a rencontré est l’affichage des stations selon le déplacement de l’utilisateur, une première idée était de calculer en utilisant les coordonnées la surface rectangle de la carte visualisée par l’utilisateur et puis afficher les stations localisées dans ce rectangle, mais cette méthode a été changé et remplacé par le calcul du centre de la map à chaque déplacement de la map et puis créer une condition qui affiche que les stations localisées dans ce rayon.

6. Temps de développement par tâche

Les tâches réalisées n’ont pas été effectuées dans un ordre chronologique, je travaillais sur chaque tâche jusqu'à ce que la tâche soit dans un niveau stable puis je commence une autre tâche et refais la même chose. Au fur et à mesure je reviens travailler sur les tâches qui ont été initiées et les faits évoluer.

- Initier la page Map en affichant la carte d' OpenStreetMap (2H)
- Créer la page Table ou on peut afficher les stations dans un tableau (3H)
- Créer la page Graph ou on peut visualiser deux graphes qui montrent la moyenne des prix des carburants dans toutes les stations, et un deuxième qui montre l’évolution des prix de ces derniers. (3H)
- Etablir la connexion avec le backend (2H)
- Parser les données reçues depuis le backend et les adapter afin de pouvoir les utiliser dans les trois pages. (5H)
- Adapter l’affichage des stations sur la carte en fonction de la position de l’utilisateur. (1H)

7. Code

7.1. Composant qui me semble élégant

src > components > JS table.jsx > ...

```jsx
  return (
    <React.Fragment>
      <TableRow className={classes.root} >
        <TableCell>
          <IconButton aria-label="expand row" size="small" onClick={() => setOpen(!open)}>
            {open ? <KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
          </IconButton>
        </TableCell>
        <TableCell component="th" scope="row">
          {row.adresse}
        </TableCell>
        <TableCell align="right">{row.ville}</TableCell>
        <TableCell align="right">{row.id}</TableCell>
        <TableCell align="right">{row.latitude}</TableCell>
        <TableCell align="right">{row.longitude}</TableCell>
      </TableRow>
      <TableRow>
        <TableCell style={{ paddingBottom: 0, paddingTop: 0 }} colSpan={6}>
          <Collapse in={open} timeout="auto" unmountOnExit>
            <Box margin={1}>
              <Typography variant="h6" gutterBottom component="div">
                Prix Carburant
              </Typography>
              <Table size="small" aria-label="purchases">
                <TableHead>
                  <TableRow style={{ backgroundColor:'#211E1E' }}>
                    <TableHeaderCellNested>Nom</TableHeaderCellNested>
                    <TableHeaderCellNested>Prix</TableHeaderCellNested>
                    <TableHeaderCellNested align="right">Date</TableHeaderCellNested>
                  </TableRow>
                </TableHead>
                <TableBody>
                  {row.prix.map((historyRow) => (
                    <TableRow key={historyRow.nom}>
                      <TableCell component="th" scope="row">
                        {historyRow.nom}
                      </TableCell>
                      <TableCell>{historyRow.valeur}</TableCell>
                      <TableCell align="right">{historyRow.maj}</TableCell>
                    </TableRow>
                  ))}
                </TableBody>
              </Table>
            </Box>
            <Box margin={1}>
              <Typography variant="h6" gutterBottom component="div">
                Services
              </Typography>
              <NumberList numbers={row.services.service} />
            </Box>

          </Collapse>
        </TableCell>
      </TableRow>
    </React.Fragment>
  );
}
```

Je pense que le code dans le composant “table.jsx” est élégant, car lors d’une première étape j’avais afficher les tableau d’une autre manière ou on avait pas des lignes cliquables mais ou on avait une liste déroulant dans la cas prix et services qui permet d’afficher la liste des ces derniers, mais le code n’était pas élégant car on affichait des listes dans des “<TableCell>” et même l’affichage du tableau n’était pas optimal. Mais maintenant on a un tableau qui est inclus dans le tableau principal ce qui rend le code plus lisible et compréhensible et aussi facile à modifier tout en gardant un affichage optimal.

7.2. Composant qui mériterait une amélioration

Le code que je pense qu'on pouvait améliorer et la partie dans le fichier "App.jsx" où on fait l'appel au backend. Dans cette partie on récupère la liste des stations par le backend puis on commence le traitement des données. Vu que dans le fichier reçu depuis le backend la liste des prix contient plusieurs prix pour le même carburant dans des dates différentes alors ce fichier n'est pas vraiment adapté pour la Map et le tableau alors il faut filtrer ce tableau et garder que les prix les plus récents des carburants. Par contre pour le graphe on peut utiliser ces données, ce qui nous a poussé à réaliser deux tableaux. Mais je pense qu'on pouvait créer des fonctions dans chaque composant et faire le traitement adapté dans le composant lui-même.