

Hadrien Bonato–Pape

Rapport Projet React

Programmation Web : Client Side

Identifiant GitHub : *Hadrien-Bonato-Pape*

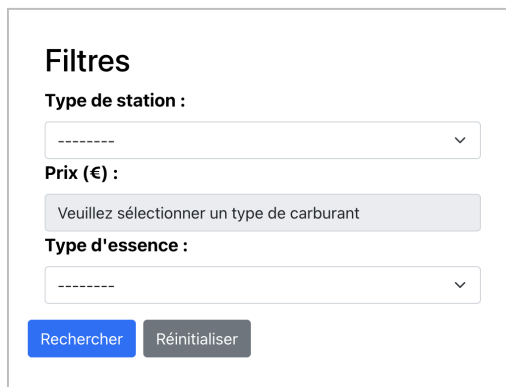
Adresse du profil GitHub : <https://github.com/Hadrien-Bonato-Pape>

Tâches effectuées

Appel à l'API

Ma première tâche a été de créer une fonction permettant d'interroger l'API. Et ainsi de récupérer la liste des stations. La fonction est "modulable", elle prend en argument des options de filtrage.

Composant filtres



Filtres

Type de station :
----- ▾

Prix (€) :
Veuillez sélectionner un type de carburant

Type d'essence :
----- ▾

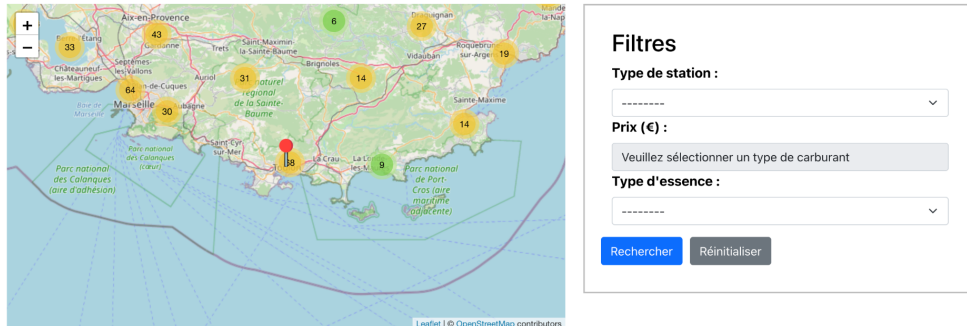
Rechercher Réinitialiser

J'ai ensuite réalisé un composant/fonction permettant de paramétrer des filtres. L'objectif étant de filtrer la liste des stations. Pour ce faire, on passe en paramètres de notre requête *http GET* les filtres souhaités, comme : le type de station (standard ou d'autoroute), le type d'essence (SP98, SP95...), le prix maximal correspondant à un type d'essence...

On utilise ce composant pour la carte et pour la liste des stations. J'ai bien sûr adapté les filtres à la carte et à la liste.

Un exemple d'intégration ci-dessous :

Map selection

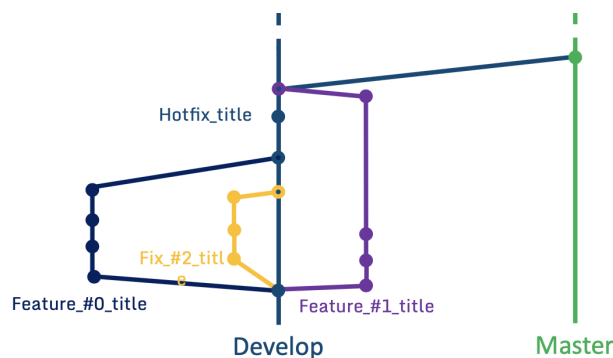


Contexte pour stocker les filtres

Afin de passer nos paramètres de filtrage au travers de l'application nous avons choisi d'utiliser un contexte qui stocke un objet "*filtre*". Celui-ci contient nos paramètres de filtrage à appliquer.

Stratégie Git

Notre stratégie Git est assez simple. Notre branche "main" contient la dernière version de l'application, une version stable. La branche "*develop*" a elle la version en cours de développement, elle est stable mais toujours en cours de développement, on ne push pas dessus directement. A partir de la branche "*develop*" nous créons des branches dédiées à une feature. Une fois le développement de cette nouvelle fonctionnalité terminé, nous effectuons une pull request vers la branche "*develop*". Et pour finir, nous supprimons la branche.



Solutions choisies

J'ai choisi d'utiliser un contexte pour stocker le filtre, parce qu'il était probable que nous utilisions les filtres à de nombreux endroits. Cependant, nous aurions pu tout à

fait utiliser les props, mais il aurait fallu faire quelques aménagements pour modifier l'utilisation du composant filtre. L'avantage ici est que si nous le souhaitons, il serait tout à fait possible de garder le même filtre entre la carte et la liste. Nous ne l'avons pas implémenté car nous préférons quand cela se remet à 0.

Difficultés rencontrées

J'ai eu quelques difficultés à comprendre le fonctionnement du contexte. Mais j'ai débloqué la situation à force de recherches.

Temps de développement / tâche

Appels à l'API : 4h

Composant filtres : développement sur plusieurs jours.

Contexte pour stocker les filtres : plusieurs heures, fonctionnement repris à de multiples reprises avec le composant filtres.

Code

Code élégant

Je qualifierais d'élégante ma fonction "MapParameters". Elle embarque le code pour paramétrer les filtres et les mettre à jour dans mon contexte.

Nous avons plusieurs variables d'état qui sont principalement utilisées dans le formulaire :

```
const [road, setRoad] = useState("");  
const [price, setPrice] = useState(0);  
const [fuel, setFuel] = useState("");
```

Elles nous servent à enregistrer les input du formulaire.

Le formulaire comporte deux select et un input qui attend des nombres (pour le prix). Il ne peut être renseigné que si un type d'essence a été choisi.

```
<Form.Group>  
  <Form.Label>Prix (€) : </Form.Label>  
  { (fuel !== "") ?  
    <Form.Control min="0" className="form-input" type="number" value={price}  
    onChange={ (e) => setPrice(e.target.value)} /> :  
    <Form.Control min="0" className="form-input" type="text"  
    disabled="disabled" value="Veuillez sélectionner un type de carburant" /> }  
</Form.Group>
```

Une fois le formulaire rempli il faut cliquer sur le bouton "rechercher", qui lance la fonction "search". Elle va venir mettre à jour le contexte avec un nouvel objet filter de la classe "filterModel".

Hadrien Bonato–Pape

```
const search = () => {  
  let filter = new FilterModel(road, price, fuel);  
  filter.checkFilters();  
  filterContext.updateFilter(filter);  
}
```

Lors de l'initialisation nous vérifions que les données renseignées sont bonnes.

Une fois l'objet créé, nous mettons à jour le contexte avec le nouveau filtre. Le contexte mis à jour permet alors aux composants consommant ce contexte d'avoir le filtre à jour. Et donc la carte et la liste affichent uniquement des données filtrées sur les critères choisis.

Code à améliorer

J'aurais pu améliorer le fonctionnement de mon contexte, en utilisant des “*props*” éventuellement.