POLYTECH NICE SOPHIA | UNIVERSITÉ CÔTE D'AZUR

Programmation Web Client-Side

EIIN927

Rapport Projet React

Anthony CHOQUARD

Année universitaire 2020-2021

Identifiant Github :

AnthonyChoquard

Tâches effectuées :

Personnellement, je me suis occupée de la carte en général.
Ceci comprend la récupération des données nécessaires puis l'instanciation des différents composants en utilisant du lazy loading. Mais aussi, l'affichage des deux cartes (régionale et départementale), les interactions avec celles-ci incluant dans un premier temps un switch pour passer d'une carte à l'autre. De plus, afin de filtrer les informations sur la map j'ai ajouté un système de slider pour filtrer la semaine que l'on souhaite visualiser, une coloration suivant le ratio de personne positive par rapport la population totale de la région ou de département, et enfin une info-bulle détaillant la région ou le département survolé et le nombre de cas positif pour la semaine choisie.

Stratégies employés pour la gestion des versions avec Git :

Pour ce qui est de la stratégie d'organisation pour le repository github, nous avons établi une branche dev sur laquelle nous avons principalement travaillé sur ce projet afin de ne push sur la branche main uniquement la version final, propre et complète de notre solution pour qu'elle soit facile d'accès aux personnes externes au groupe souhaitant récupérer le projet. De même nous sommes restés sur une stratégie simple de push et de stash avant de pull. M'étant principalement occupé des composant Map, MapDep et MapReg, j'étais la personne qui modifiait le plus ces composants donc je pouvais attendre de bien terminer une feature et d'avoir vérifier qu'elle soit fonctionnelle pour la push, ce qui m'as permis d'éviter les conflits majoritairement et aussi de faire des commit un peu plus conséquent contenant beaucoup de changements.

Solutions choisies :

- ### Plugin pour le graph carte

Pour ce qui est du plugin utilisé pour le dessin de la carte et les interactions avec celle-ci j'ai décidé d'utiliser la librairie *react-svg-map*.En effet parmis tout ceux que j'ai pus tester, ce plugin était le plus simple d'utilisation, avec un github et un readme complet m'ayant permis de comprendre comment fonctionnait le plugin. De plus, le plugin est simple d'utilisation et il fut facile de le modifier afin d'obtenir une carte satisfaisant nos attentes, quant à ce que nous souhaitions inclure dans l'application. Enfin le plugin inclut une carte découpée en département et une autre découpée en régions ce qui nous intéressait fortement de part le fait que nous avions assez de données pour remplir les deux cartes ce qui représentait l'information supplémentaire a mettre a disposition des futurs utilisateurs de notre solution.
Il existe de nombreuses alternatives pour les plugins notamment.

Premièrement MapBox permettant de créer des cartes sur un logiciel tierce en téléchargeant directement les données dessus puis en incluant la carte dans le site, mais qui par conséquent nécessite de retourner sur le logiciel pour ajouter des données ou changer les datasets pour générer de nouveaux la carte, la ou notre solution permet de changer les données dans le back ce qui se répercute directement sur la carte quie st chargée dynamiquement à chaque fois que l'on lance le site.
De plus il y avait aussi la possibilité d'ajouter une instance de Google Map ou encore un GeoChart, à laquelle nous aurions ajouter les informations souhaitée sous forme de couleurs ou de cluster, cependant l'instance de google map requiert une api key et donc un compte développeur google (ce qui n'est pas spécialement un problème en soit car j'en possédait déjà un pour de précédent projets) mais aussi et surtout car la façon d'intégrer les données pour les cartes était pensé pour les intégrer en dur dans le code et sous un format précis, contrairement à la librairie retenu pour notre solution. En effet la librairie choisi a conscience selon la carte fournie des différents départements ou région et nous permettait de comparer de manière dynamique soit les id des département afin de les colorés un par uns selon les données du serveur, soit l'abréviation du nom de la région après un petit traitement de notre côté dans un fichier csv afin de faire correspondre les données du serveur et celle du librairie.

- ### Solution pour filtrer les informations à l'écran

Pour le filtrage des informations nous avons décidé d'intégrer deux filtres, le choix de la carte (régionale/départementale) via un switch, car c'est l'options qui nous paraissait la plus intuitives et de la même manière un slider pour le choix de la semaine car en plus de sembler plus intuitif permet à l'utilisateurs de voir en temps réel les données changer à l'écran alors qu'il fait glisser le curseur sur le slider ce qui permet de visualiser l'évolution de la situation plus facilement semaines après semaines.
Il existe de très nombreuse manière de filtrer de permettre aux utilisateurs de filtrer les données affichées, notamment via deux bouton pour passer d'une map a l'autres, cependant nous trouvions qu'un switch serait plus approprié comme pour le mode sombre, indiquant deux uniques cas possible et donc soit l'un soit l'autres actuellement affiché. Ici cela apporte une information en plus qu'est celle du type de carte affiché selon l'état du switch. De plus, le slider aurait pu être remplacé par un menu de sélection déroulant contenant toutes les semaines affichables. Cependant le menu déroulant ne permettait pas de voir le changement au fur et à mesure du changement de la valeur obligeant à ouvrir le menu sélectionné observer les informations et recommencer. Le slider permet de ce fait, d'observer graduellement le changement des couleurs et autres données sur la carte.

Difficultés rencontrées :

- ### Choix de la librairie

Dans un premier temps il a fallu trouver la bonne librairies pour utiliser la carte ce qui fut une tâche assez complexe. En effet j'ai passé mes deux premiers cours dédiés à ce

projet a chercher une librairies satisfaisant les attentes que nous avions pour notre carte. Cependant il a fallu tester les librairies pour voir si elles proposais bien ce que nous cherchions et j'ai donc tester dans un premier temps d'utiliser MapBox qui ne répondait pas à mes attentes car les cartes était généré via un site tierce de manière statique et ne pouvait donc pas récupérer les données du serveur pour se mettre à jour dynamiquement. Ensuite nous avons testé GeoChart de google qui malheureusement proposait de remplir la cartes avec des données fournies en dur dans le code et sous un format qui nous contraignait trop. Enfin nous avons testé la librairie D3JS, celle ci était la plus complète et la plus proche de nos attentes, mais aussi celle qui nous a pris le plus de temps a testé avant de nous rendre compte qu'il ne nous était pas possible de passer les données récupérées du back et donc impossible d'utiliser la carte la rendant inutile dans notre projet.
Trouver la bonne librairie fut fortement chronophage mais après les avoir toutes testées j'ai fini par tomber sur la librairie *react-svg-map* qui offrait une version simplifiée de l'utilisation de "heatmap" permettant la coloration de différente zone indépendamment. De plus cette librairie propose une version départementales et une version régionale de la carte de la france ce qui nous intéressait beaucoup, et surtout la librairie possède une variable récupérée automatiquement correspondant au différente location selon la carte actuellement utilisée (les différents département et les différentes régions) me permettant de comparer les identifiants avec les données que nous possédions dans le serveur afin de colorer la map tel que nous le souhaitions. Enfin les SVGMap (composant accessibles par la librairies) possèdent différents paramètres qui correspondent à des fonctions permettant de lancer des événement lorsque la carte est survolée par la souris par exemple récupérant la location du lieu survolé et l'ayant permis de remplir une info bulle contenant les informations liée à la location.

- **La récupération des données**

La seconde difficultés fut la gestion de la récupération des données nécessaires à la création de la carte. En effet, celles-ci n'étant pas instantanément chargées et devant être récupérées dans le serveur, la carte, initialement, s'initialisait avant que les données n'arrivent, la rendant inutilisable car la carte restait noire n'ayant pas les données nécessaires à la coloration.
Pour contrer cela j'avais initialement ajouter des tests afin d'initialiser la carte seulement quand les données serait disponibles puis nous avons ajouté à notre solution du lazy loading, afin de n'importer les composant lié à la carte seulement quand les données du serveur seront récupérées dans leur intégralité, et permettant par la même occasion d'afficher un chargement renseignant à l'utilisateur que le temps de chargement est normal et qu'il n'y a pas d'erreur.

Temps de développement par tâches

- Choix de la librairie en les testant une à une (~7h)
- Apprentissage de react-svg-map et affichage de la première carte (~3h)
- Récupération des données nécessaires pour la carte (~1h)
- Traitement des données et coloration de la carte en fonction de celles-ci (~3h)
- Ajout et remplissage de l'infobulle (~30min)
- Ajout du slider pour filtrer la semaine affichée sur la carte (~2h)
- Ajout de la carte régionales, test d'affichage et changement de l'architecture pour séparer les deux cartes en deux sous composants (~4h)
- Ajout du switch pour passer d'une carte à l'autre (~30min)
- Calcul de la date en fonction du numéro de la semaine (~30min)
- Utilisations de Lazy loading pour le chargement des données des cartes (~1h)

Code

- **Fonctions élégantes ou optimales**

```
const handleLocationMouseOver = (event) => {
  const pointedLocation = getLocationName(event);
  for (let i = 0; i < currentDisplayedData.length; i++) {
    if (currentDisplayedData[i][type] == (type == "reg" ? RegId[0][getLocationId(event)] : getLocationId(event))) {
      setPointedLocationData(currentDisplayedData[i].P);
    }
  }
  setPointedLocation(pointedLocation);
}

const handleLocationMouseOut = () => {
  setPointedLocation(null);
  setTooltipsStyle({ display: 'none' });
}

const handleLocationMouseMove = (event) => {
  const tooltipStyle = {
    display: 'block',
    top: event.clientY + 10,
    left: event.clientX - 100
  };
  setTooltipsStyle(tooltipStyle);
}

const getLocationClassName = (location, index) => {
  UpdateDisplayedData();
  for (let i = 0; i < currentDisplayedData.length; i++) {
    if (currentDisplayedData[i][type] == (type == "reg" ? RegId[0][location.id] : location.id)) {
      let ratio = currentDisplayedData[i].P / currentDisplayedData[i].pop;
      if (ratio <= 0.0003125) return `svg-map__location--heat-1`;
      else if (ratio <= 0.000625) return `svg-map__location--heat0`;
      else if (ratio <= 0.00125) return `svg-map__location--heat1`;
      else if (ratio <= 0.0025) return `svg-map__location--heat2`;
      else if (ratio <= 0.003125) return `svg-map__location--heat3`;
      else return `svg-map__location--heat4`;
    }
  }
}
```

Sur cette image on retrouve les fonctions fournies au composant SVGMap afin de gérer le comportement de la carte (l.48 à 85 dans DepMap.js)

- getLocationName s'occupe de colorer la carte selon le ratio de personnes contaminées, pour un département ou une région donnée dont on récupère les données liées grâce au paramètre "location" et les informations du serveur.
- handleLocationMouseOver détermine le lieu survolé par la souris set set les données dans des hook useState afin de les réutiliser lors de l'affichage de l'info bulle.
- handleLocationMouseOut supprime l'info bulle et réinitialise les informations contenues dans celle-ci.
- handleLocationMouseMove réaffiche l'info bulle à l'emplacement de la souris afin que celle-ci suive la souris.

Je trouve ce code élégant et optimal grâce à l'utilisation des hook useState pour le partage des informations à afficher entre les différentes fonctions ainsi que pour l'affichage ou non de l'info bulle en changeant le style du pop-up. De plus, l'utilisation de ternaire permet de définir la manière de trier les données, différentes selon le type de carte (départementale ou régionale). Enfin la coloration de la carte est décidée par le ratio et la fonction getLocationName qui renvoie une classe SCSS déterminant la couleur du lieu.

- ## Composant qui mériterait une optimisation

Pour ce qui est du code qui mériterait une optimisation je souhaiterais améliorer l'architecture des sous composants car beaucoup de fonctions présentes (notamment les quatres fonctions citées précédemment) dans MapDep et MapReg sont assez similaires et pourraient être factorisées en un seul composant. Je ne suis malheureusement pas parvenu à le faire car l'initialisation des deux cartes ne se faisait pas en parallèle et il résidait toujours une des deux cartes qui restait intégralement noire quoi qu'il arrive et dont les données n'étaient pas téléchargées empêchant la coloration et le remplissage de l'info bulle