Faouzi ABDELJAOUED

Groupe: AR
Repo: https://github.com/TiFosi/prog-web
Github: https://github.com/TiFosi

Tâches effectuées:

- Implémenter le composant NationalSituation: front, api call: J'ai implémenté le composant NationalSituation, qui affiche des indicateurs sur le Covid-19 par rapport à toute la France à partir du résultat d'un appel vers un API externe.
  API utilisé: https://github.com/florianzemma/CoronavirusAPI-France

- A partir d'un composant du framework AntD, j'ai implémenté un spinner pour afficher le chargement des composants.

- Dockerize l'application: mettre l'application en version docker: un environnement développement qui supporte le rechargement automatique et une version qui construit l'application à partir d'un build.

- J'ai implémenté le composant Chart qui représente les informations en une courbe.

- J'ai implémenté en backend les routes "/taux-incidence/std-quot-reg/:id" et "/taux-incidence/std-quot-dep/:id" pour récupérer respectivement le taux d'incidence par région et par département.

- Implémenter la fonctionnalité du toggle Dark mode.

- Implémenter une carte Mapbox GL qui affiche les hospitalisations par département.

- Implémenter la fonctionnalité de détection de la position de l'utilisateur et de zoomer sur elle dans la carte.

- Implémenter la fonctionnalité de détecter le thème du système d'exploitation de l'utilisateur et lui proposer de changer le mode dans l'application.

Git workflow:

- Notre stratégie dans la gestion des versions du code était de garder la branche master pour les versions stables de notre code et d'avoir la branche develop comme notre repère, d'où on pull tous. Pour le push, on a décidé de protéger develop et de développer dans des branches en local. Ensuite, une fois qu'on veut envoyer notre code

on push la branche et on ouvre une pull request pour la merger dans develop. Le merge dans develop passe par l'approbation de l'autre développeur. Cette stratégie nous a permis de garder develop à l'abri des conflits. Mais en approchant vers la deadline du projet, et pour gagner du temps, on a terminé par pushé directement sur develop surtout qu'on est seulement deux personnes sur ce projet et on pouvait en discuter des modifications avant de les pusher.

Solutions choisies:

- Front:
  - AntD: On a utlisé ce framework parce qu'il est recommandé avec le développement avec React. C'est une librairie de composants React UI minimalistes et on a trouvé la documentation très intuitive. On aurait pu utiliser Boostrap ou Material UI, on a opté d'ailleurs pour Bootsrap au début mais on a voulu manipuler des composants React et c'est AntD qui a attiré notre attention.
  - Highcharts: Pour la modélisation des graphiques, on a utilisé finalement la librairie Highcharts au lieu de Recharts avec laquelle on a implémenté un exemple avant d'en renoncer. On a trouvé Highcharts plus complet dans les types de graphes qu'elle propose et d'une certaine façon très encapsulée, elle gère tout ce que vous lui lancez dans un objet "options".
  - Mapbox GL: On a utilisé cette librairie pour afficher une carte interactive. On aurait pu utiliser Google maps mais ce choix a été fait parce que les exemples qu'on a essayé de suivre ont été faits par Mapbox GL.
- Back:
  - Express.js: On aurait pu développer la partie serveur directement en Node.js mais on voulait tirer bénéfice des fonctionnalités de ce framework comme les routeurs et l'architecture MVC.
  - Mongodb Atlas: On a choisi d'utiliser une solution cloud pour stocker la base de données pour notre application au lieu de se soucier de la partie installation et maintenance de la base de données habituelle.

Difficultés rencontrées:

- Responsiveness dans le Composant NationalSituation: J'ai eu du mal à afficher correctement les 4 indicateurs selon le type d'écran. Pour remédier à ce problème j'ai utilisé react-columns au lieu des rows et colonnes de AntD.
- Dans la modélisation des données de taux d'incidence dans une courbe, j'ai essayé d'utiliser Recharts et BasicLine chart de Highcharts mais le rendu n'était pas au top parce que le nombre des points à afficher est trop grand donc j'ai utilisé un autre type qui est StockChart qui permet d'échantillonner les points par période.

- Après l’implémentation avec succès de la carte avec react-map-gl en développement, je voulais tester le rendu avec le build mais la carte ne s'affiche pas. Il s’est avéré que c’est un problème avec webpack et j’ai trouvé la solution est de rétrograder à une autre version du package react-map-gl.
- Une autre difficulté rencontrée avec la responsiveness, cette fois-ci avec la carte. La dimension initiale de la carte ne change pas lorsqu’on agrandit ou réduit l’écran. La solution était d’ajouter un listener à l’écran et un hook qui permet de redimensionner la carte à chaque fois que la dimension de l’écran change.

Un composant que j’ai écrit et qui me semble élégant:

NationalSituation.js: Ce composant récupère les données d’un API externe et affiche les informations de chaque indicateur dans une section dédiée. J’ai implémenté un sous composant DataItem.js dans NationalSituation.js pour encapsuler l’affichage de l’indicateur et le réutiliser en passant juste le nombre total et le texte à afficher dans props. J’ai implémenté aussi dans ce composant un spinner qui s’affiche pendant le chargement des données de l’API et j’ai utilisé pour ça les hooks useState et useEffect. A mon avis, ce n’est pas un composant très complexe en comparaison avec Map ou Chart qui dépendent de bibliothèques externes mais la façon avec laquelle il a été fait est élégante.

Un composant que j’ai écrit et qui mériterait une optimisation ou amélioration:

Map.js: Ce composant est très complexe, déjà dans sa version actuelle m’a demandé beaucoup d'efforts pour se documenter sur la bibliothèque react-map-gl et essayer de corriger les bugs liés à l’affichage. J’ai utilisé 4 types de hook dans ce composant.
Afin de tirer le maximum de cette librairie, il mérite des améliorations comme:
- Afficher plusieurs indicateurs dans la même carte.
- Afficher des bulles par région est zoomer sur chaque bulle pour afficher les bulles des départements qui appartiennent à cette région.
- Améliorer la détection de la position utilisateur en détectant aussi le département et afficher les données lui sont liées.
- Ajouter un événement onHover pour afficher plus d’informations par rapport à une bulle.