Emilie Tombuyses
Virgil Limongi
WEB2 P2021

Choix technologiques

Agora est une plateforme en ligne en collaboration avec le gouvernement abordant le thème de l'écologie. Elle se veut accessible à tous les citoyens français souhaitant contribuer à l'amélioration de l'environnement par des missions mensuelles à remplir sur notre plateforme.

Nous avons choisi de développer cette plateforme en utilisant la méthode de client-side rendering car parmis toutes les méthodes de rendering existantes, nous avons jugé que c'était le moyen le plus adapté pour consommer un minimum de ressources tout en conservant un minimum de fluidité.

- La méthode de server-side rendering n'a pas été retenue car le principe de création de page depuis zéro pour chacune d'elle nous a semblé consommé beaucoup de ressources et de bande passante inutile. Le temps mis pour chaque requête nous a aussi semblé un frein au niveau de la fluidité du site.

- Le static rendering a également été écarté car ne permet pas de générer des données dynamiques qu'utilisent notre site.

- Enfin l'universal-rendering consomme plus de ressources que le client-side rendering. C'est pourquoi nous avons choisi d'utiliser du client-side rendering

Nous avons choisi d'utiliser un framework Js pour la création de notre site car il se présente sous la forme d'un dashboard. Par conséquence, utiliser un framework js tel que React, Vue ou encore Angular au sein de notre projet nous a permis de faciliter le développement des différents modules ci-trouvant. React a été préféré en tant que framework de développement car bien qu'étant comparable en termes de performance aux autres frameworks comme Vue ou Angular, il s'agit du plus connu et du plus utilisé sur le marché aujourd'hui.

Au niveau de nos librairies de développement, nous avons choisi de travailler avec styled-components ainsi que node-sass. Styled-components afin de pouvoir regrouper le css et la logique d'un component au sein d'un même fichier, ce qui nous a permis d'avoir recours plus facilement à la méthodologie de l'atomic design dans notre structure de développement en définissant des components atomes, molécules et organismes dont l'apparence est directement modifiable au sein du fichier du component. Node-sass nous a lui servi à définir la configuration plus générale de l'application, incluant un reset css, des variables de couleurs, une configuration de fonts et certaines mixins.

Souhaitant rendre le site accessible à tous au niveau de son utilisation et de son ergonomie, nous avons opté pour la simplicité et l'efficacité, c'est pourquoi nous n'avons utilisé qu'une librairie pour le contenu du site. Il s'agit de react-d3-radar, développée à partir de la librairie d3js connue pour ses graphiques en SVG.

Nous avons fait le choix de cette librairie car les svg comportent plusieurs avantages. Les svg, par définition sont vectoriels et scalables, ils s'adaptent donc à toute taille d'écran. Leur poids est relativement léger et leur affichage ne demande pas beaucoup de ressources. Cette librairie est également plus légère que celle de d3-js.

Dans le choix de nos images et icônes, nous avons essayé d'utiliser un maximum de svg quand cela était possible pour les mêmes raison citée ci-dessus, de plus le png possède un poids plus important que le svg. Cependant, nous souhaitions avant tout conserver une cohérence graphique pour tous les utilisateurs, c'est pourquoi les icônes des tâches sont des images png (équivalent svg non trouvé) afin qu'elles apparaissent de la même façon peut importe le device utilisé.

Notre application étant destinés à tous les profils technologiques, nous trouvions important de faire de notre application une PWA pour faciliter son accessibilité sur mobile. Pour cela nous avons complété le manifest.json et également implémenté un service-worker permettant à l'application de fonctionner hors-ligne avec les données gardées en mémoire (pas besoin d'actualiser sa consommation d'électricité sur le mois toutes les heures).

Etant donné que nous avions choisi de développer une PWA, nous trouvions important de développer une application en mobile first totalement responsive utilisant en moyenne 3 breakpoints (tablette, desktop et desktop large).

Nous avons également fait en sorte d'avoir un maximum de compatibilité browser afin d'assurer une bonne accessibilité du site. Le site est ainsi accessible sur edge, chrome, safari, opera et firefox. Il n'est cependant pas accessible sur IE car React n'est pas totalement compatible avec celui-ci.

Concernant les polices, nous avons fait le choix d'utiliser le CDN de Google fonts car les fonts téléchargées sont optimisées en terme de poids en fonction de notre device et de notre browser. Google fonts réduit également le nombre de caractères de la fonts à ceux qui seront affichés (Font subsetting) pour encore plus d'optimisation. Source : https://www.tunetheweb.com/blog/should-you-self-host-google-fonts/

Enfin, pour renforcer l'aspect écologique abordé par notre application, nous avons également eu recours à la technique du code splitting. Au lieu de charger tout le js du site depuis un seul fichier à l'arrivée sur le site, le client ne va ainsi charger que le fichier js de la page demandée, évitant ainsi de charger du code inutile de pages qui ne seront peut être pas visitées par l'utilisateur.