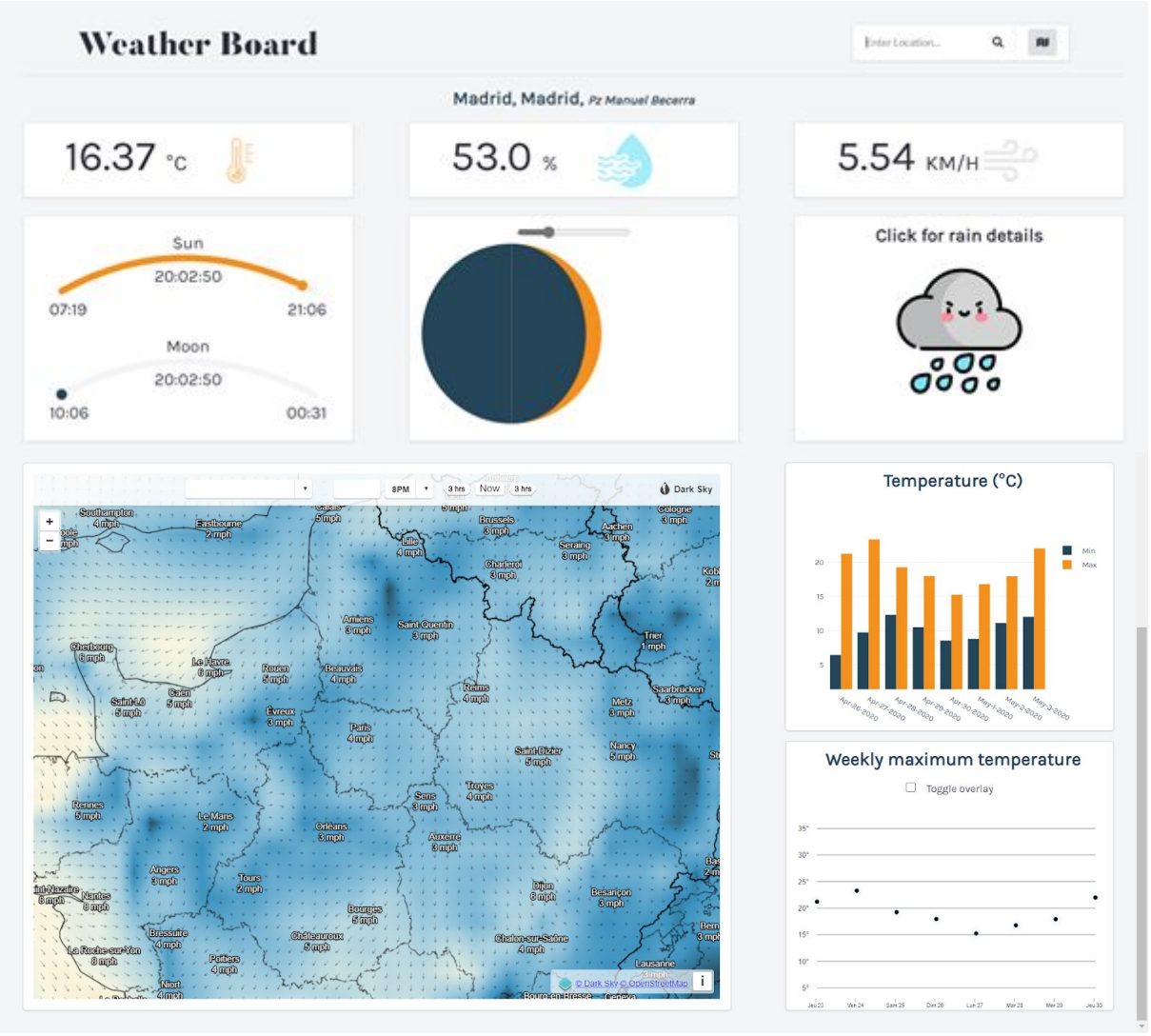
Weather Board

Rapport de projet



Enguerrand De Smet

Margaux Vaillant

Nicolas Liénart

Antoine Libert

IMAC
INGÉNIEUR
ESIPE

Lien vers site web

https://dashboard.ingenus.ovh/

Genèse du projet

Réflexion sur le concept

Après avoir constitué l'équipe, nous avons brainstormé ensemble sur le sujet du dashboard. Enguerrand a suggéré que nous travaillions à partir d'un jeu qu'il développe actuellement. Il était en train d'y mettre en oeuvre un programme de deep learning capable d'apprendre à jouer en simulant des parties. A partir de ce programme, nous pourrions générer des données à exploiter afin de mesurer les progrès de l'IA par génération.

Toutefois, le développement du programme prenant plus de temps que prévu, nous avons choisi de nous replier sur le second thème que nous avions envisagé : la météo. Nous voulions avant tout profiter de cet exercice pour développer une diversité d'éléments visuels constituant le dashboard. La météo nous permettrait en effet d'exploiter une certaine richesse de visuels : à la fois des cartes, des graphiques, des représentations du soleil, un pluviomètre, etc. Par ailleurs, c'est un thème sur lequel il existe un large panel de données fournies par des APIs en libre accès.

Réflexion sur la technologie

Nicolas et Antoine avaient déjà travaillé à l'occasion de leur projet tuteuré sur le framework Vue.js, Enguerrand pour son portfolio et Margaux a souhaité découvrir un framework. C'est pourquoi dès la constitution de l'équipe on a retenu Vue afin de permettre d'approfondir les connaissances de certains, et de donner l'occasion de le découvrir pour d'autres.

Présentation du concept

Notre dashboard vise donc à proposer un aperçu général des conditions météorologiques pour une zone géographique déterminée. Nous récupérons d'une part la localisation du client via son adresse IP et d'autre part l'ensemble des données météo (température, humidité, vent, …) associées à cette localisation via API. Nous pouvons interagir avec ces données en choisissant manuellement une autre localisation que la sienne, soit en tapant le nom d'une ville, soit en se déplaçant sur une carte. Nous récupérons les coordonnées géographiques associées à la ville et actualisons l'ensemble des données du dashboard en fonction de cette demande

Scénario d'usage

Lorsque l'on arrive sur le dashboard, on peut tout d'abord autoriser l'accès à sa localisation. Les informations pourront ainsi être adaptées en fonction de l'emplacement de l'utilisateur. On peut le vérifier facilement grâce à l'adresse présente en haut du dashboard. On peut ensuite découvrir les différents éléments du dashboard, se déplacer sur la carte, y afficher les informations concernant le vent. On peut ensuite survoler les graphiques pour y afficher des informations plus précises par endroits. Sur le graphique du dessous, le bouton Toggle Overlay permet de voir l'animation qui s'effectue normalement lors de l'actualisation des informations. Concernant les informations lunaires, on voit la phase de la lune actuellement, mais il est possible grâce au slider de voir le visuel en fonction de d'autres phases lunaires. Enfin, en cliquant sur le petit nuage, une pop up s'affiche montrant les taux d'humidités des prochains jours grâce à des pluviomètres. Un slider permet de jouer sur les baromètres.

Maintenant, il est possible d'indiquer des noms de villes dans la barre de recherche en haut (comme Madrid, Tokyo, votre ville natale...) et les informations vont alors s'actualiser. Avec l'icône map juste à côté, une carte s'affiche et on peut se déplacer et cliquer pour sélectionner une nouvelle localité. Une nouvelle fois les informations s'actualisent. Enfin, on peut tester le responsive du site en modifiant la largeur d'affichage.

Journal de progression

Mise à niveau

La première étape était de s'assurer que nous avions toutes les connaissances nécessaires pour évoluer ensemble dans le développement en Vue.js sans trop de difficultés. Ceux d'entre nous qui connaissaient peu le framework prirent l'initiative de s'informer sur les tutos et les conseils des autres. Puis, ceux qui l'avaient déjà expérimenté ont développé un premier « fil conducteur » : une base de projet présentant la logique des composants, un appel d'API et l'exploitation de ses données dans un graphe. De la même façon que nous l'avions fait en TP avec HyperApp, nous avons parcouru ce fil conducteur ensemble lors d'une réunion afin de nous assurer que l'ensemble était suffisamment bien compris par tous.

Phase d'expérimentation

Une fois cette base posée, nous sommes tous de notre côté partis dans l'expérimentation de technologies permettant de développer des éléments de data-visualisation : vue-graphJs, Plotly, D3.js, ou des créations personnelles. Nous avons ensuite fait une réunion pour découvrir les expérimentations des autres et les techniques qu'ils ont utilisées. Nous avons enfin évalué les éléments que nous voulions garder dans le dashboard final, en fonction de leur intérêt pour le projet mais en privilégiant également les plus intéressants techniquement.

Uniformisation

Nous avons par la suite décidé de l'organisation visuelle générale du dashboard en fonction des éléments que nous avions sélectionnés. Nous y avons ajouté des éléments que nous avons moins travaillés comme la carte récupérée via Iframe depuis l'API de DarkSky, car ce sont des éléments que nous trouvions nécessaires dans la cohérence générale du dashboard.

Le choix de l'API pour récupérer les données météo a évolué durant le projet en fonction de nos besoins : nous utilisons d'abord Open Weather pour les températures, la géolocalisation pour le coucher du soleil et la phase de la lune, l'API DarkSky pour la carte. Nous avons finalement fusionné ces récupérations

d'informations en récupérant (presque) l'ensemble des données sur l'API DarkSky afin de limiter le nombre de requêtes à effectuer.

Pour finir, nous avons défini quelques éléments de charte graphique avec pour objectif d'uniformiser la mise en forme des éléments. Nous nous sommes ensuite réparti les sections du dashboard pour faciliter le travail en parallèle.

Clean du code

Dernière phase, une fois le dashboard ressemblant à ce que nous souhaitions, nous avons tous fait un tour global du code afin d'identifier :

- Les améliorations possibles dans la structure et l'organisation du code dans les composants.
- Les bonnes gestions des erreurs en cas de mauvaise réponse d'une API.
- Les optimisations dans le traitement de données et la performance (lorsqu'elles étaient simples à faire).
- L'élégance du code, dans sa lecture (noms variables, indentations…)
- Etc.

Zoom sur des éléments techniques et justifications

Le choix de Vue.js

Au-delà de nos connaissances préalables, nous trouvions ce framework particulièrement adapté à la création de dashboard. Sa gestion des composants au sein d'une même page (Single File Component) nous a permis de très facilement découper les éléments du projet. Cela a également facilité notre travail de groupe car il était ainsi plus simple de travailler sur des modules séparés sans attendre le résultat des autres.

Une synthèse rapide de la météo locale

Nous voulions privilégier une visualisation rapide des données météo à l'endroit où se trouve l'utilisateur, car c'est en majorité ce que souhaitent les

personnes recherchant des informations météo sur internet. On peut ainsi, juste en arrivant sur la page, avoir les infos qui nous concernent. Puis venir ensuite vers un usage plus spécifique en se renseignant sur les données météo d'une autre localité.

Concernant les données que nous avons privilégiées, nous nous sommes focalisés sur les 3 questions les plus demandées concernant la météo : Quelle est la température ? Va-t-il pleuvoir ? Y a-t-il du vent ? Notre objectif était de pouvoir répondre à ces questions très simplement à la fois concernant aujourd'hui, mais aussi les 7 prochains jours.

Deux manières de choisir une localité

Nous tenions à proposer deux manières intuitives de rechercher la météo dans une ville précise. Au début du projet, nous récupérions les données sur l'API via le noms de certaines villes directement, ce qui nous obligeait de limiter la recherche à une sélection de nom que l'API connaissait. Mais nous avons ensuite utilisé une autre API pour déterminer à partir d'un nom de ville donné des coordonnées polaires. Cela qui nous a permis de laisser l'utilisateur choisir la ville qu'il souhaitait via un champ libre.

Parallèlement à cela nous avons développé un moyen de choisir la localisation en se déplaçant sur une carte directement. Là où la saisie d'un nom de ville correspond à un utilisateur qui sait spécifiquement ce qu'il recherche, cette seconde option permet de plutôt se balader par curiosité pour découvrir la météo de région que l'on ne connaît pas nécessairement.

Difficultés rencontrées

Nous avons créé plusieurs modules de visualisation customisés, en vanilla JS où via D3.js, car en tant qu'IMAC nous avions des idées borderline qui ne trouvaient pas de candidat préexistant. On a donc dû mettre les mains dans la les SVG avec leur système de placement pas toujours évident, la conception de courbes de béziers. Ou encore de gérer l'évolution de la donnée au cours du temps, et effectuer des transformations en conséquence qui sont parfois chaînées. D'ailleurs nous avons pu mettre à profit le fonctionnement asynchrone et utiliser les Promises pour passer d'état en état.

Post-mortem

Nous avons tous, dès le début du projet, tenu à y développer nos connaissances dans la maîtrise de framework Vue.js et des différentes solutions permettant de créer des éléments de data-visualisation. D3.js par exemple est un véritable standard dans ce domaine, et son approche relativement bas niveau et mettant l'accent sur la fluctuation de la donnée offre de très nombreuses possibilités de mise en forme. Ce fut d'ailleurs le cas avec la visualisation pluviométrique où on a pu créer un systèmes de gauges millimétrées aisément, la seule limite est notre imagination. Avec le graphe des températures maximum, c'est plutôt pour l'animation que D3.js a été intéressant, car bien qu'étant un graphique classique on l'a mis en forme et animé exactement comme on l'imaginait, on n'a pas eu de contraintes imposées par la librairie.