Benchmark du Front-end - Projet OMS IA

Introduction

Dans le cadre du projet pour l'OMS, nous avons réalisé un benchmark comparatif des différentes solutions front-end qui s'offrent à nous. Mon objectif était de comparer plusieurs technologies pour voir laquelle serait la plus adaptée au projet de plateforme de prédiction pandémique.

Les solutions comparées

J'ai choisi ces 6 combinaisons parce qu'elles reviennent souvent dans les discussions et correspondent aux technologies mentionnées dans notre webographie :

1. React.js avec D3.js et Material-UI
2. React.js avec Chart.js et Bootstrap
3. Vue.js avec D3.js et Vuetify
4. Vue.js avec Chart.js et Tailwind CSS
5. Angular avec Plotly.js et Angular Material
6. Svelte avec Chart.js et Tailwind CSS

Pour chaque solution, nous avons regardé quatre aspects importants : l'accessibilité, la facilité de développement, l'écosystème autour de la techno, et l'adaptabilité aux besoins du projet.

Test de l'accessibilité

L'accessibilité est importante à évaluer vu que l’application doit respecter les standards WCAG 2.1.

React avec D3.js et Material-UI

Material-UI a un support d'accessibilité avec plein de composants qui gèrent automatiquement les attributs ARIA.

React avec Chart.js et Bootstrap

Bootstrap a quelques composants accessibles par défaut, mais Chart.js demande du travail supplémentaire pour bien gérer l'accessibilité des graphiques.

Vue.js avec D3.js et Vuetify

Vuetify fournit des efforts sur l'accessibilité, et Vue.js permet d'ajouter facilement les attributs ARIA nécessaires.

Vue.js avec Chart.js et Tailwind

Tailwind ne fournit pas de composants prêts, donc tout l'aspect accessibilité dépend de nous. Mais de ce fait, on a un contrôle total. Avec un peu d'effort, on peut faire du très bon travail.

Angular avec Plotly et Angular Material

Angular Material est probablement la bibliothèque la plus mature en termes d'accessibilité. Plotly.js gère bien l'accessibilité des graphiques nativement.

Svelte avec Chart.js et Tailwind

Svelte est encore jeune et l'écosystème d'accessibilité n'est pas encore au niveau des autres.

Facilité de développement

Là c'est vraiment important pour nous étudiants ! J'ai chronométré le temps que ça me prenait pour faire un petit tableau de bord avec chaque solution.

Vue.js avec Chart.js et Tailwind

La documentation de Vue est excellente, Chart.js est super simple à utiliser, et Tailwind permet de styler rapidement. Clairement le meilleur sur ce critère.

Vue.js avec D3.js et Vuetify

D3.js demande plus de temps d'apprentissage, mais une fois qu'on comprend les concepts, c'est puissant. Vuetify accélère le développement avec ses composants prêts.

React avec Chart.js et Bootstrap

React demande un peu plus de configuration au début, et il faut comprendre les concepts de state et props. Mais une fois lancé, ça va bien.

React avec D3.js et Material-UI

La combinaison React + D3.js n'est pas évidente, il faut bien comprendre comment faire cohabiter le Virtual DOM de React avec les manipulations directes du DOM de D3.

Angular avec Plotly et Angular Material

Angular a beaucoup de concepts à maîtriser (services, modules, dependency injection...), et la configuration initiale est lourde.

Svelte avec Chart.js et Tailwind

La compilation de Svelte est vraiment impressionnante, mais l'écosystème est encore petit.

Écosystème et support

React

La communauté est énorme, il y a des tonnes de ressources, tutorials, et bibliothèques disponibles.

Vue.js

La documentation officielle est très claire et avec plein d'exemples.

Angular

Une grande communauté et beaucoup de ressources, mais parfois la documentation est un peu dense.

Svelte

La communauté grandit mais reste petite. L'écosystème manque encore de maturité.

Adaptabilité

Pour notre projet, nous avons regardé la capacité de chaque solution à créer des visualisations complexes, à être responsive, et à pouvoir évoluer facilement.

React avec D3.js et Material-UI

On peut créer n'importe quel type de graphique avec D3.js, et Material-UI assure une interface cohérente. C'est extensible et maintenable.

Vue.js avec D3.js et Vuetify

Dans la même veine, avec l'avantage que Vue.js est plus simple à maintenir. Très bon aussi.

Vue.js avec Chart.js et Tailwind

Peut-être moins puissant pour des visualisations très poussées, mais largement suffisant pour notre projet et très maintenable.

React avec Chart.js et Bootstrap

Correcte pour des besoins standard.

Angular avec Plotly et Angular Material

Complexe à faire évoluer.

Svelte avec Chart.js et Tailwind

Sympa mais l'écosystème limité pose question pour l'évolutivité.

Comparaison et analyse

Après tous ces tests, voici comment nous voyons les choses :

Les meilleurs pour de l'accessibilité sont clairement React + D3.js + Material-UI et Angular + Plotly + Material. Ces solutions ont vraiment tout ce qu'il faut pour respecter les standards WCAG 2.1 sans trop se prendre la tête.

Pour la facilité de développement, Vue.js avec Chart.js et Tailwind gagne haut la main. Svelte est sympa aussi mais l'écosystème plus petit pose problème.

L'écosystème, c'est du React partout. Vue.js se défend bien, mais React reste le dominant. Angular a les ressources mais c'est parfois indigeste.

Pour l'adaptabilité, les solutions avec D3.js dominent, mais elles demandent plus d'expertise technique.

Ma recommandation

Après tous ces tests, je recommande **React avec Chart.js et Tailwind CSS**.

Cette solution n'est peut-être pas la meilleure partout, mais elle n'a aucune faiblesse. Elle excelle en facilité de développement, a de bonnes performances, un écosystème solide, et une accessibilité correcte qu'on peut améliorer.

Pourquoi pas les solutions avec D3.js qui semblent plus puissantes ? Parce que nous sommes une équipe de 4 avec 19 heures de préparation. La courbe d'apprentissage de D3.js est vraiment raide, et on risque de perdre trop de temps à comprendre les concepts au lieu de développer des fonctionnalités pour les utilisateurs finaux.

Pourquoi avoir remplacer Vus.js par React, comme dit plus tôt, nous n'avons que 19 heures de préparation, notre équipe maîtrise déjà React au détriment de Vus.js qui est nouveau pour l'ensemble de l'équipe. Nous avons peur de perdre trop de temps sur l'apprentissage de la techno c'est pour cela que nous avons choisi React plutôt que Vus.js.

Avec React + Chart.js + Tailwind, on peut être productifs très rapidement. Chart.js nous donnera des graphiques propres et suffisamment accessibles pour 90% de nos besoins. Tailwind nous permettra de créer une interface responsive rapidement sans se prendre la tête avec du CSS custom.

C'est la solution la plus pragmatique pour notre contexte.