Rapport Exercice Frontend SpaceX

Fabio ETOILE

Sommaire :

Introduction :

Ce projet a été réalisé dans le cadre d'un test technique visant à évaluer mes compétences en développement front-end avec Vue.js 3, TypeScript et Tailwind CSS. L'objectif était de concevoir une application web permettant d'afficher et de filtrer les lancements de SpaceX en utilisant l'API publique SpaceX v5.

Pour mener à bien ce projet, j'ai utilisé :

- Vue.js 3 avec la Composition API pour structurer l'application.
- TypeScript pour un typage strict et une meilleure maintenabilité du code.
- Fetch API pour récupérer les données des lancements via requêtes HTTP asynchrones.
- Tailwind CSS pour assurer une mise en page rapide et moderne.
- Gestion d'état avec ref et computed pour gérer les données dynamiques et les filtres.

L'application devait intégrer plusieurs fonctionnalités clés, comme l'affichage du prochain lancement, la possibilité de filtrer les lancements passés, et un modal interactif affichant des détails enrichis (image, vidéo, article, charge utile, etc.).

L'objectif était de proposer un rendu fonctionnel et soigné tout en respectant les contraintes de temps imposées.

Choix techniques et optimisations :

1️⃣ Structure et organisation du projet

L'application suit une structure modulaire et organisée selon les bonnes pratiques Vue.js :

- Composants dans src/components/ :
- ListeLancements.vue → Gère l'affichage et le filtrage des lancements.
- ModalLancement.vue → Affiche les détails d'un lancement dans une fenêtre modale.
- ProchainLancement.vue → Présente le prochain lancement avec un compte à rebours dynamique.
- Assets stockés dans src/assets/ (images de fond, logos, rapport).
- CSS global avec Tailwind
- Configuration et scripts dans vite.config.ts, tsconfig.json, package.json, etc.

2️⃣ Utilisation de Vue.js 3 avec Composition API

L'application est basée sur la Composition API pour une meilleure modularité et lisibilité du code.

- ref() et computed() sont utilisés pour gérer les états réactifs (ex : filtres et sélection de lancement).
- onMounted() permet de charger les données API dès le montage du composant.
- watch() suit les changements de sélection et met à jour dynamiquement les informations affichées.

💡 Optimisation : Filtrage optimisé via computed() pour éviter un re-render inutile.

3️⃣ Récupération des données via Fetch API

Les données des lancements sont chargées dynamiquement depuis l'API **SpaceX v5** en **asynchrone** (async/await).

- **Endpoint utilisé** :
    - *https://api.spacexdata.com/v5/launches/past*
    - *https://api.spacexdata.com/v5/launches/next*
- **Gestion des erreurs** : try/catch pour éviter les crashs en cas de requête échouée.

4️⃣ Gestion des performances

1) **Filtrage côté client** pour limiter les appels API.
2) **Tri des données** dès la récupération pour éviter des calculs à chaque re-render.
3) **Suppression des doublons** (ex : Starlink) pour un affichage pertinent.
4) **Lazy Loading des vidéos** pour ne charger l'iframe YouTube que lorsque l'utilisateur clique.

5️⃣ UI/UX : Design avec Tailwind CSS

L'application utilise Tailwind CSS pour une mise en page simple et efficace :

- Composants réactifs (flex, grid, space-x-4).
- Boutons et select améliorés (hover:bg-gray-700, rounded-lg, shadow-md).
- Animations fluides (transition, ease-in-out).

Problèmes rencontrés :

Lors du développement de l'application, plusieurs problèmes ont été rencontrés. Tout d'abord, l'API SpaceX v5 n'était pas toujours à jour(dernier commit datant de 2022), ce qui a entraîné des incohérences dans certaines données, notamment pour les articles et les détails. Ensuite, un bug d'affichage est apparu lors de l'activation des vidéos YouTube pour les lancements échoués, provoquant une duplication du fond d'écran que je n'ai pas réussi à régler à tant. De plus, le manque de connexion stable a rendu le travail à domicile plus compliqué, notamment pour tester l'application en temps réel et récupérer les données de l'API avec de bonnes performances.

Résultat final et axes d'amélioration

Le projet final est pleinement fonctionnel et répond aux exigences initiales, avec une interface claire et une navigation fluide. L'application permet d'afficher les prochains lancements, de filtrer les lancements par statut et de consulter des informations détaillées via un modal interactif.

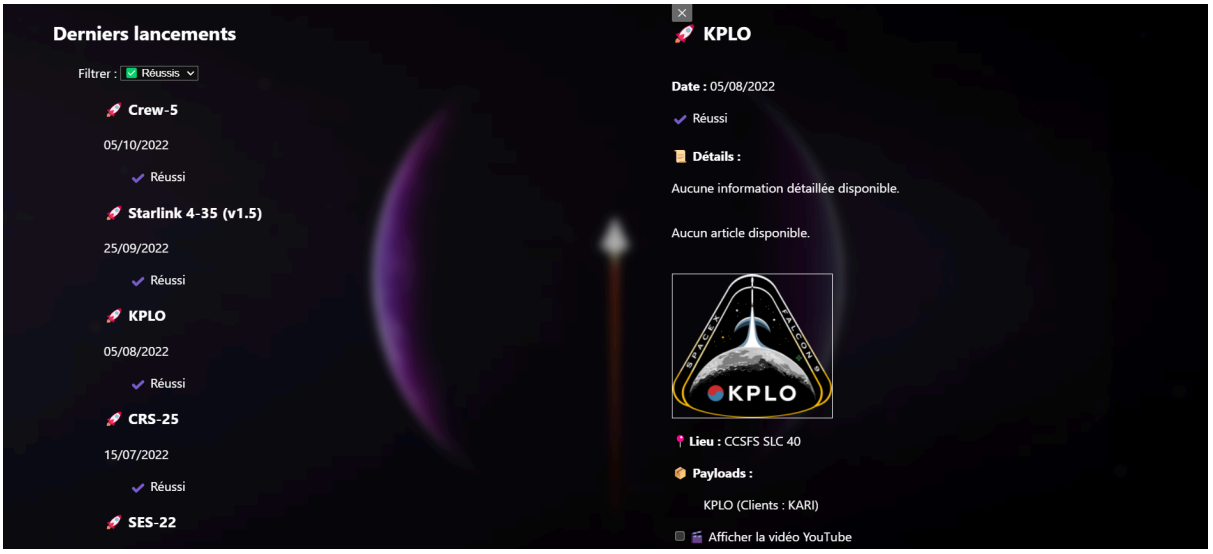
Cependant, certaines améliorations pourraient être envisagées :

- Correction du bug de duplication du background lorsqu'une vidéo est affichée dans les lancements échoués.
- Affinement de la mise en page avec Tailwind CSS pour un rendu plus structuré et esthétique, car un peu trop simpliste à mon goût.
- Utilisation d'une API plus à jour pour garantir l'affichage des dernières données SpaceX.

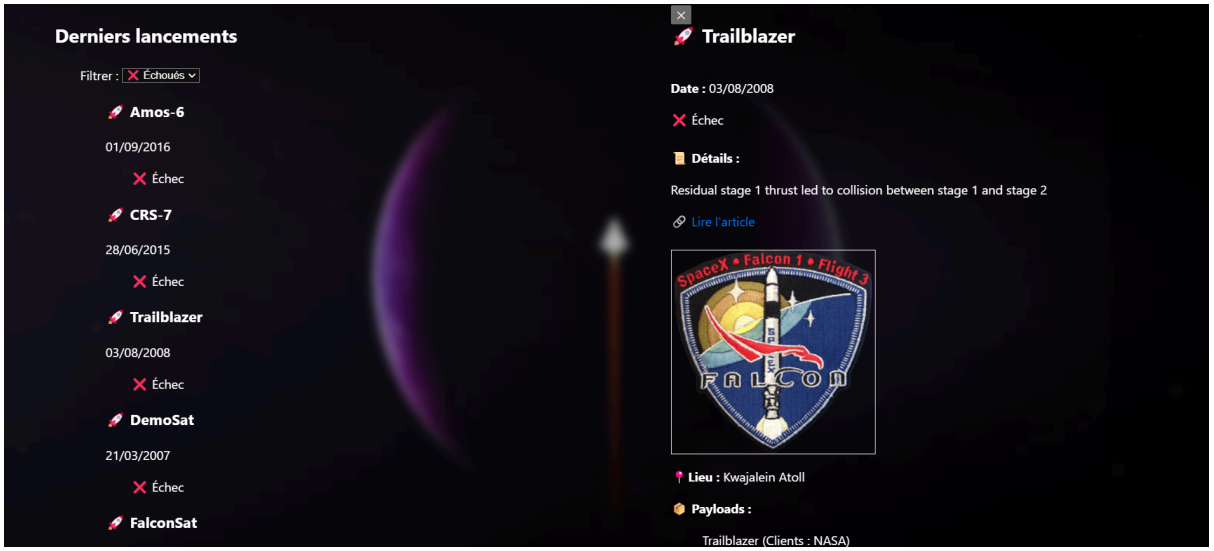
Vous trouverez ci-dessous des screen du travail effectué :

Page principal avec tous les lancements :

Page avec les lancements réussies :



Page avec les lancements échouées :



Conclusion personnelle:

Ce projet a été une expérience enrichissante, me permettant de découvrir et d'expérimenter des technologies que je n'avais jamais utilisées auparavant, telles que Vue.js, TypeScript et Tailwind CSS. Bien que ces frameworks m'étaient inconnus au départ, leur syntaxe et leur logique de fonctionnement m'ont semblé assez familières, facilitant ainsi leur prise en main.

J'ai particulièrement apprécié travailler sur cette application, car elle m'a offert une expérience pratique avec ces outils et m'a permis d'approfondir mes compétences en développement frontend. Cependant, la réalisation du projet n'a pas été sans difficultés. Ma connexion instable a parfois rendu le travail compliqué. De plus, j'ai dû gérer d'autres projets académiques et des contrôles en parallèle, ce qui a ajouté une contrainte de temps non négligeable, mais aussi je n'ai pas pu exploiter tout mon temps, car j'ai un CDI étudiant le week-end de 14 heures, qui fait que je n'ai pas pu consacrant 100% de mon temps à ce projet.

Malgré ces péripéties, je suis satisfait du résultat final. L'application fonctionne comme prévu, et même si certaines améliorations restent possibles, elle répond aux exigences demandées.