Rapport du Sprint 2

Création d'une page web responsive avec HTML, Tailwind CSS et JavaScript

Abdelkarim Sadiki

Sprint 2 – Développement Frontend

13 février 2026

1 Objectifs du Sprint

Les objectifs fixés pour ce sprint étaient les suivants :

— Créer une page web responsive sans framework JavaScript.

— Utiliser une structure HTML sémantique claire.

— Exploiter Tailwind CSS pour le responsive design et le mode sombre.

— Gérer le préchargement et le lazy loading des médias.

— Mettre en place une navigation moderne (mega menu, burger menu).

— Prendre en compte le SEO et l'accessibilité.

2 Choix de la Maquette

Pour ce projet, une maquette Figma a été sélectionnée parmi celles proposées. Ce choix a permis de travailler sur plusieurs composants (header, hero section, sections de contenu, formulaire, footer) tout en respectant une architecture claire et modulaire.

La maquette a servi de référence visuelle pour :

— la disposition des éléments,

— la hiérarchie visuelle,

— la cohérence graphique,

— l'expérience utilisateur globale.

3 Implémentation Technique

3.1 Structure HTML et Sémantique

La structure HTML a été pensée pour respecter les standards du web :

— Utilisation des balises sémantiques (`header`, `nav`, `main`, `section`, `footer`).

— Hiérarchisation correcte des titres (`h1` à `h6`).

— Association correcte des labels et des champs de formulaire.

— Utilisation d'attributs `alt` pour les images.

Cette approche améliore à la fois l'accessibilité et le référencement naturel (SEO).

3.2 Stylisation avec Tailwind CSS

Tailwind CSS a été utilisé comme framework CSS principal :

— Gestion du responsive design via les classes `sm`, `md`, `lg`.

— Implémentation du mode sombre avec `dark:`.

— Mise en place d'animations légères et de transitions.

— Utilisation des utilitaires d'aspect ratio pour les médias.

Cette approche utilitaire permet un développement rapide, lisible et maintenable.

3.3 Images Responsives et Performance

Afin d'optimiser les performances :

— Le lazy loading est activé pour les images non critiques.

— Les ressources importantes sont préchargées lorsque nécessaire.

Ces optimisations améliorent le temps de chargement et le score Lighthouse.

3.4 JavaScript et Interactivité

JavaScript est utilisé uniquement pour ajouter de l'interactivité :

— Gestion d'un Slider.

— Controler le formulaire (Emails, tel ...) et afficher des message d'erreurs.

— Au clic sur 'View All Posts' > récupérer en Ajax d'autres items et les appender.

Le code JavaScript est organisé de manière modulaire et exécuté après le chargement du DOM.

3.5 Accessibilité et SEO

Une attention particulière a été portée à :

— l'utilisation des attributs ARIA lorsque nécessaire,

— la navigation au clavier,

— les meta tags SEO (description, viewport, Open Graph),

— la compatibilité multi-navigateurs.

4 Résultat et Liens

Le projet final est accessible en ligne et son code source est disponible sur GitHub :

— Démo en ligne : https://stage-void-iqih6c7s7-karims-projects-6c4b9e12.vercel.app/

— Dépôt GitHub : https://github.com/krimoSD/Stage-VOID/tree/main/Sprint2

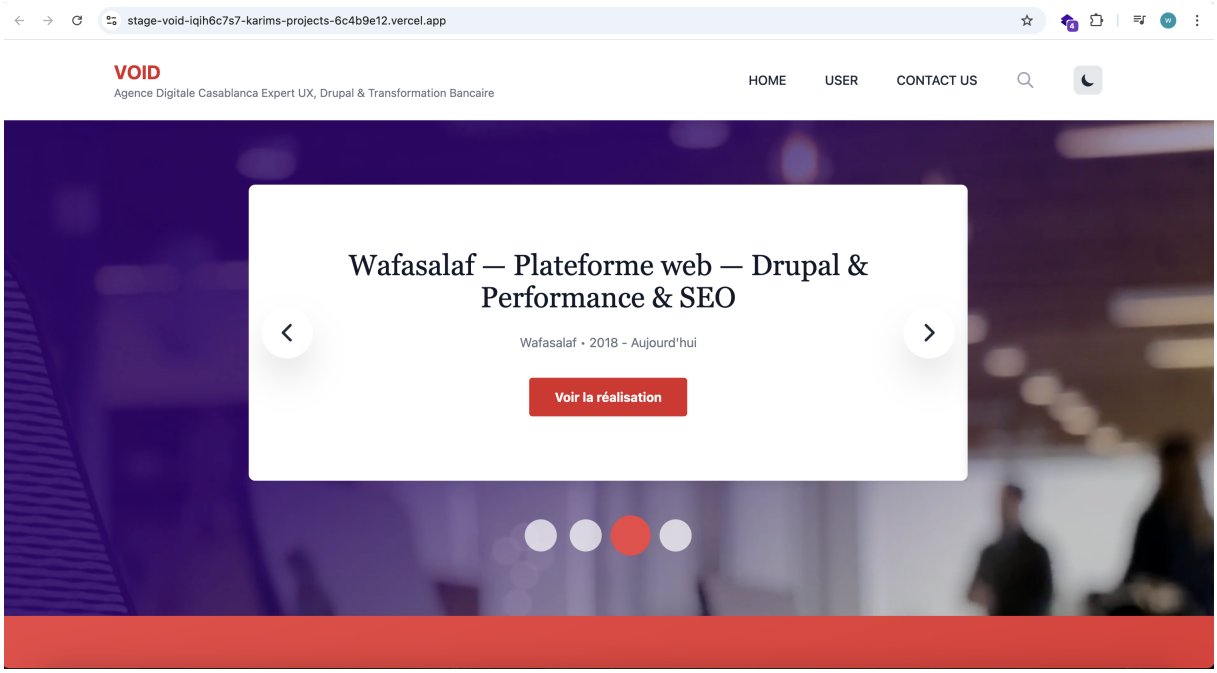

FIGURE 1 – Le site web hoster en vercel

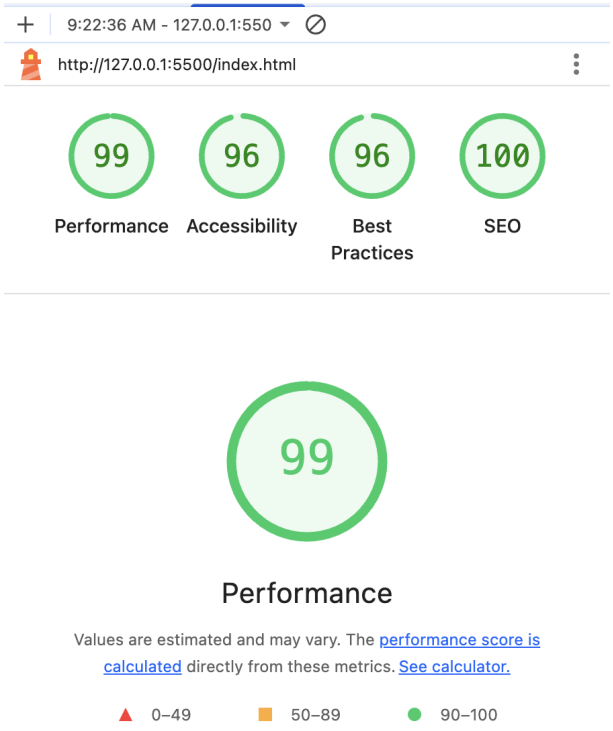

FIGURE 2 – Resulta de test lighthouse