Dans le but d’améliorer mon projet initialement élaboré avec HTML, CSS et JavaScript, j’ai opté pour le cadre du framework Nuxt.js basé sur Vue.js et notamment pour mieux organiser le code et pour développer des fonctions plus avancées.

## Mise en œuvre de Nuxt.js

J’ai dans un premier temps installé Node.js pour pouvoir faire fonctionner Nuxt, et dans un second temps, j’ai mis en place mon projet Nuxt avec npx create-nuxt-app, et j’ai produit certains choix utiles afin de tenir compte de mes besoins :

Gestionnaire de packages : npm est ma préférence (yarn étant également proposé).

Moteur de template : j’ai opté pour le choix HTML.

Modules de Nuxt : j’ai rajouté Axios pour simplifier les requêtes et PWA pour donner plus de performance à mon application.

Linting and CI (Continuous Integration) : ESLint a mon choix afin de garder le code propre et bien structuré. Je savais pas ce que c’est mais, j’ai toujours du mal à savoir à quoi ça sert mais je penses que c’est pour éviter les faute de syntaxe etc

Rendu mode : j’ai retenu le rendu côté serveur (SSR) pour faire que le site soit plus rapide à charger pour l’utilisateur.

Ces choix déterminants ont eu un impact direct sur le déroulement du projet et lui ont apporté de la fluidité, de la structure.

Migration des Fichiers HTML, CSS et JavaScript préexistants

Ensuite, j’ai désarticulé mes fichiers HTML en composants Vue, et je les ai placés dans des dossiers components ou pages. A titre d’exemple, j’ai converti index.html en ChatArea.vue et login.html en Login.vue placés dans pages afin d’être accessibles par /login.

## Autres interrogations et difficultés rencontrées

J'ai rencontré au cours de cette migration un bon nombre d’interrogations et de difficultés :

Chemins d’accès non valables : Tout au long du travail, plusieurs chemins de fichiers n’étaient pas les bons, et j’ai donc dû les corriger en utilisant @/ et ~/.

Modules absents : Il a fallu procéder à l’installation de certains modules dans le but d’éviter d’avancer avec des erreurs de compilation.

Problèmes de styles avec le scoped : J’ai également compris que scoped empêchait l’application de certains styles au niveau global ce qui m’a poussé à l’enlever pour certains styles.

Erreurs de linting : J’ai dû renommer certains fichiers sous certaines situations (Login.vue en UserLogin.vue par exemple) pour respecter les bonnes pratiques.

Rendu côté serveur (SSR) : J’ai compris que le rendu côté serveur permettait de rendre l’affichage et la vitesse de chargement plus intéressants pour l’utilisateur.

Pour conclure, cette migration a permis de mieux structurer mon projet, d’ajouter plus facilement les nouvelles fonctionnalités les plus avancées, tout en rendant le projet plus rapide et plus simple à maintenir. Même si le passage à Nuxt a parfois été difficile, cela est très bénéfique pour la performance et l’organisation de mon projet.

## Push github

npx husky install

npx husky set .husky/pre-commit "exit 0"

// soucis avec husky

La convention de message de commit consiste à structurer le message avec un **type** (feat, fix, etc.) suivi d'une **courte description** du changement, par exemple : feat: ajout de la fonctionnalité de chat. J’ai eu des soucis, mais une fois le message de commit arrangé, le commit est passé.