Dossier Remise de mi-Parcours projet Web

Boudrouss Réda

Groupe 7 n°28712638

Sorbonne Université
France

Contents

Outils utilisés

Ce projet est réalisé avec les frameworks et outils suivants :

- Typescript pour le développement. Typescript est équivalent au Javascript mais avec des types statiques. Cela permet de développer plus rapidement et de trouver plus facilement les erreurs.
- React pour la création de l'interface utilisateur coté client.
- Next.js pour la gestion de la backend et de la génération du site web. Nextjs offre la possibilité aussi de prégénérer certain composant coté serveur pour améliorer les performances. Il s'occupe aussi du routing et de la génération des pages.
- Prisma pour la gestion de la base de données. Prisma est un ORM qui permet de gérer la base de données, ses migrations, ses requêtes et est particulièrment adapté au Typescript car il permet de faciliter le typage.

Structure générale du projet

Au root du projet vous trouverez plusieurs dossiers *(ignorons les fichiers de configuration)* :

- `src` : contient le code source du projet
- `public` : contient les fichiers statiques du projet
- `prisma` : contient les fichiers de configuration de la base de données (et la base de données si sqlite est utilisé). Le fichier `schema.prisma` contient la définition de la base de données.

Concernant le dossier `src` il contient plusieurs sous-dossiers :

- `app` : contient le coté ""client"" de l'application, toutes les pages et composants sont dans ce dossier.
- `pages/api` : contient le coté serveur et plus précisément l'API.
- `helper` : contient des fonctions utilitaires.

Coté client (React)

Un des avantages de Next13+ et le dossier `app` est le fait qu'il s'occupe pour nous du routing de l'application de part la structure du dossier. Par exemple le fichier `app/login/page.tsx` est la page atteignable à l'adresse `/login`. Pour plus d'information, voir la documentation de Next.js

Structure du dossier `app`

Dans le dossier `app`, tout les dossiers serv à déterminer le chemin de la page. Excepté le dossier `components` qui contient les composants réutilisables.

`components`

Les components utilisés dans la pages sont : (leurs props seront détaillés si nécessaire, pour une liste complète voir l'image ci-dessous)

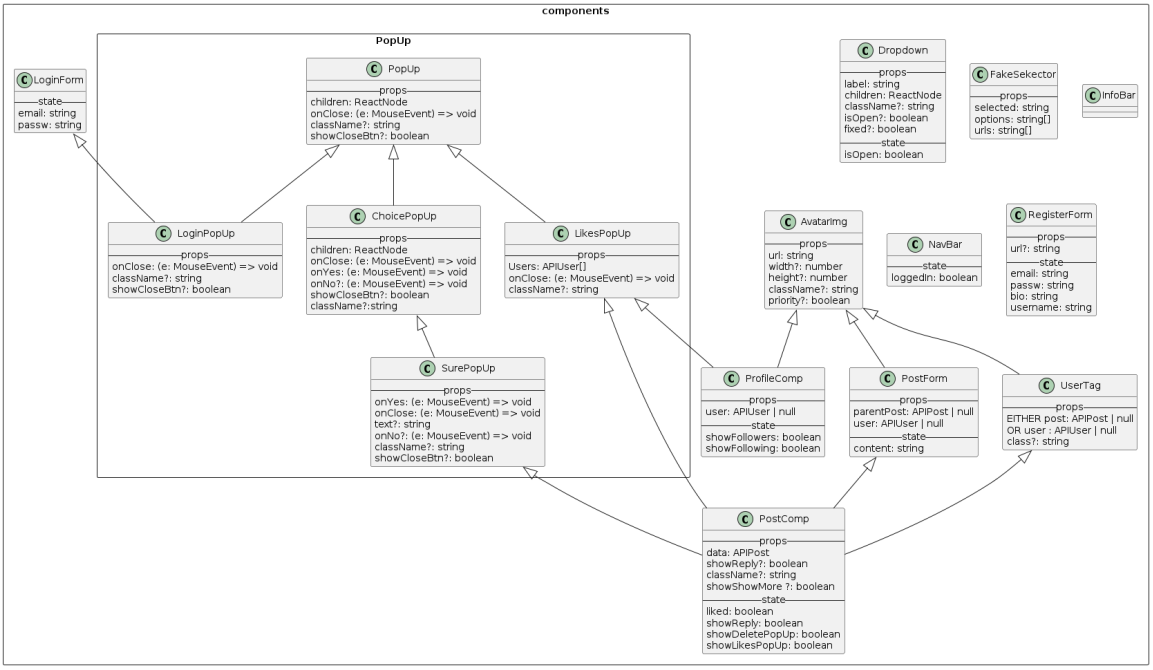

Figure 1: figure des dépendances des components

- `AvatarImg` : Le composant de l'avatar de l'utilisateur. Il prend en paramètre l'url de l'image et la taille. Il est possible de passer une classe css pour le modifier.
- `Dropdown` : Liste déroulante, affiche/cache les enfants quand on clique dessus. Prend en paramètre le label affiché, les composants react à afficher et l'état par défaut. (utilisé dans la page `settings`)
- `FakeSelector` : Composant qui donne l'impression d'un choix fluide mais est en réalité que des liens. Prend en paramètre les choix à afficher ainsi que leurs URLs et le choix séléctionné (Solution temporaire)
- `InfoBar`: Barre d'information affiché à droite sur la version desktop. Vide pour l'instant mais nous comptons y ajouter la bar de recherche et des recommandations.
- `LoginForm`: Formulaire de connexion. Prend rien en paramètre.
- `NavBar`: Barre de navigation affiché à gauche sur la version desktop. Contient les liens vers les pages principales de l'application et le nom de l'utilisateur connecté.

- `PopUp` : Composant qui affiche un composant prit en paramètre au milieu de l'écran. Prend en paramètre le composant à afficher et une fonction exécuté en cas de fermeture. Certain composant "hérite" de celui là.
  - `ChoicePopUp` : Un PopUp qui affiche le composant prti en paramètre suivi par un choix "oui" ou "non". Prend en paramètre les mêmes composant que `PopUp` ainsi que les fonctions à exécuter en cas de choix.
  - `SurePopUp` : hérite de `ChoicePopUp` et affiche un message de confirmation. Prend en paramètre les mêmes composant que `ChoicePopUp` ainsi que le message à afficher.
  - `LoginPopUp` : Affiche un PopUp du form de connexion. Prend en paramètre les mêmes composant que `PopUp`.
- `PostComp` : Composant d'un post. Prend en paramètre le post à afficher.
- `PostForm` : Formulaire de création de post. Prend en paramètre l'utilisateur connecté et le post parent (si il y en a).
- `ProfileComp` : Composant d'un profil. Prend en paramètre l'utilisateur à afficher. Affiché en haut dans la page `profile`.