DOCUMENTATION TECHNIQUE : STRUCTURE DU CODE

Table des matières

ARBORESCENCE :

----client

```
|_____src
        |_____assets
        |           |_____clap.png
        |           |_____hand-heart.png
        |           |_____heart.png
        |           |_____smile.png
        |           |_____thumb.png
        |
        |_____components
        |           |_____Messaging.jsx
        |           |_____ModifyUserForm.jsx
        |           |_____Navbar.jsx
        |           |_____NewPost.jsx
        |           |_____Post.jsx
        |           |_____ReactionBar.jsx
        |           |_____Scroller.jsx
        |_____pages
        |           |_____Dashboard .jsx
        |           |_____LandingPage.jsx
        |           |_____Login.jsx
        |           |_____NewHiringOffer.jsx
        |           |_____NewProject.jsx
```

| |_____OrganisationProfile.jsx

| |_____Search.jsx

| |_____Signup.jsx

| |_____Timeline.jsx

| |_____UserProfile.jsx

|_____utils

| |_____fetchData.jsx

| |_____hashPassword.jsx

| |_____regex.jsx

| |_____session.jsx

| |_____theme.jsx

|_____.env

|_____App.jsx

INSTALLATION :

git clone https://github.com/youssoufmiyad/Projet-Final-2023-2024.git

cd Projet-Final-2023-2024/client

npm run dev

FICHIERS DE CONFIGURATIONS :

- .env
- Utils/theme.jsx (gestion du theme MUI material)

OPERATIONS COMPLEXES :

Chargement des compte (App.jsx) :

A l'aide du hook « useContext », l'array contenant l'intégralié des utilisateurs est accessible depuis n'importe quel component de notre application.

Une requête est envoyé à notre API pour récupérer tout les utilisateurs (voir fonction getUsers() du fichier fetchData.jsx).

Création de compte (Signup.jsx) :

Remplissage du form :

Les valeurs correspondantes aux différent champs requis sont stockés dans des variables State.

Elles sont changés dynamiquement en fonction de la saisie utilisateur grâce à l'attribut « OnChange »

```
119                 <TextField
120                     error={
121                         (lastNameError === "") | (lastNameError.length < 1) ? false : true
122                     }
123                     helperText={lastNameError}
124                     id="filled-basic"
125                     label="nom de famille"
126                     variant="filled"
127                     onChange={(e) => {
128                         setLastName(e.target.value);
129                     }}
130                     sx={{ width: "360px" }}
131                 />
```

Envoi du formulaire :

Une fois envoyer, le form envoi une requête à notre API via la fonction « createUser » situé dans le fichier utils/fetchData.jsx

```
export const createUser = async (firstName, lastName, email, password) => {
    const hashPWD = hashPassword(password);
    const requestOptions = {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
            firstname: firstName,
            lastname: lastName,
            email: email,
            password: hashPWD,
        }),
    };
    const response = await fetch("http://localhost:8080/users", requestOptions);
    window.location.replace("../");
};
```

Connexion (Login.jsx) :

Remplissage du form : voir Inscription

Verification de l'existence d'un utilisateur :

Les valeurs saisit par l'utilisateur sont testé via regExp afin de s'assurer qu'elles soient valide. On parcours ensuite la liste d'utilisateur dans le context (voir Chargement des comptes) et on vérifie que les informations correspondent à un utilisateur.

Création de post (NewPost.jsx) :

Le component NewPost.jsx est affecté à un component Modal de la librairie MUI material. Une fois le bouton « Envoyer » pressé, une requête post est envoyé à notre API de gestion des publications (voir fonctions publishPost() dans le fichier fetchData.jsx).

Modification de l'utilisateur (ModifyUserForm.jsx) :

Passe par la fonction « modifyUser », opération géré par l'API avec une requête PATCH.

Suivre un utilisateur (fetchData.jsx) :

Géré par les différentes fonctions intitulé « follow ».