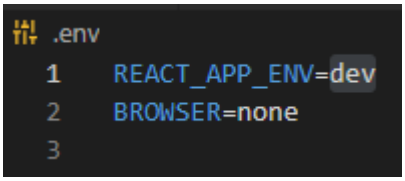
React, Node et MySql Login (Partie 8)

Création d'un Helper (fetcher) pour simplifier les requêtes (côté React)
Envoi du token d'autorisation (clé d'accès) à l'API à chaque requête (côté React)

Étape 15 : Créer un fichier .env et un fichier config
A la racine de l'appli React, créer un fichier .env (s'il n'existe pas) et y ajouter une variable d'environnement REACT_APP_ENV=dev



BROWSER=none permet de ne pas ouvrir un onglet du navigateur à chaque npm start

Puis ajouter un fichier config à la racine de src (copier-coller la config ci-dessous)

```
const config = {
    dev : {
        api: {
            url: "http://localhost:5000/",
            authorization:
"eyJhbGciOiJIUzI1NiJ9.IU1sX01PQVVHOFgpa1RrYmt1RjZdSmt5UkFPL1NELUs.6wuklfS6qogS2a
4x9p5e_c17bqfMaGVNa-x5GxtQVJ4",
        }
    },

    prod : {
        api: {
            url: "",
            authorization: ""
        }
    }

}

export default config[process.env.REACT_APP_ENV];
```

url sert à stocker l'url de base de l'API Node.js et authorization sert à stocker la clé d'accès à l'API sous la forme d'un JWT
La clé correspondante au token ci-dessus est :

```
!Ml_MOAUG8X)kTkbkuF6]JkyRAO/SD-K
```

Étape 16 : Créer un helper pour l'envoi des requêtes vers l'api
Le fichier fetcher.js est dans un sous-dossier helpers de src

src > helpers > fetcher.js > ...

```
1   import config from "../config";
2   const fetcher = {};
3
4   const request = async (endpoint, params, method = "get", body = null ) => {
5       const url = config.api.url + endpoint;
6       const options = {
7           method,
8           credentials: "include",
9           headers: {
10              Authorization: config.api.authorization,
11              "content-type": "application/json",
12          },
13          ...params
14      }
15      if(body && method !== "get") options.body = JSON.stringify(body);
16      try{
17          const resp = await fetch(url, options);
18          const json = await resp.json();
19          return json;
20      }
21      catch (error){
22          return {data: null, result: false, message: error};
23      }
24  }
25
```

Ln 1 : import de la config
Ln 2 : création de l'objet qui sera exporté
Ln 4 création d'une fonction générique (qui ne sera pas exportée)
le paramètre endpoint la route demandée à l'api
params permet d'ajouter des options à la requête
Ln 5 : l'url correspond à l'url stocké en config (http://localhost:5000) auquel on ajoute la route
Ln 6 à 14 : construction des options avec la méthode, le credentials: include (envoi automatique des cookies), les headers (authorization contient le JWT avec la clé d'accès à l'api stockée dans la config) et éventuellement des options supplémentaires reçues en dans le paramètre params
Ln 15 : si un body est passé et que la méthode HTTP n'est pas GET, nous l'ajutons aux options
Ln 16 à 23 : envoi de la requête, si tout se passe bien, notre fonction renvoie la réponse obtenue en json, sinon un objet contenant le message d'erreur est retourné.

On ajoute ensuite une fonction par méthode HTTP qui utilise notre fonction request

src > helpers > fetcher.js > ...

```
25
26  fetcher.get = async (endpoint, params = {}) => {
27      return await request(endpoint, params);
28  }
29
30  fetcher.post = async (endpoint, body = {}, params = {}) => {
31      return await request(endpoint, params, "post", body);
32  }
33
34  fetcher.put = async (endpoint, body = {}, params = {}) => {
35      return await request(endpoint, params, "put", body);
36  }
37
38  fetcher.patch = async (endpoint, body = {}, params = {}) => {
39      return await request(endpoint, params, "patch", body);
40  }
41
42  fetcher.delete = async (endpoint, body = {}, params = {}) => {
43      return await request(endpoint, params, "delete", body);
44  }
45
46  export default fetcher;
47
```

Enfin, nous simplifions l'écriture des requêtes déjà écrites dans notre appli React
Dans le AuthContext :

src > contexts > AuthContext.jsx > ...

```
9
10    useEffect(() => {
11      const doFetch = async() => {
12        const resp = await fetcher.get("auth");
13        console.log(resp);
14        setAuth(resp.data);
15      }
16      doFetch();
17    }, []);
18
```

Dans LoginScreen :

src > screen > LoginScreen.jsx > LoginScreen

```
12
13    const handleSubmit = async (e) => {
14      e.preventDefault();
15      const form = e.currentTarget;
16      const formData = new FormData(form);
17      const jsonData = Object.fromEntries(formData.entries());
18      //const body = JSON.stringify(jsonData);
19      const resp = await fetcher.post("login", jsonData);
20      console.log(resp);
21      setAuth(resp.data);
22      setAuthCookie(resp.token ?? null, {"max-age":`${60*60*24}`});
23    };
24
```