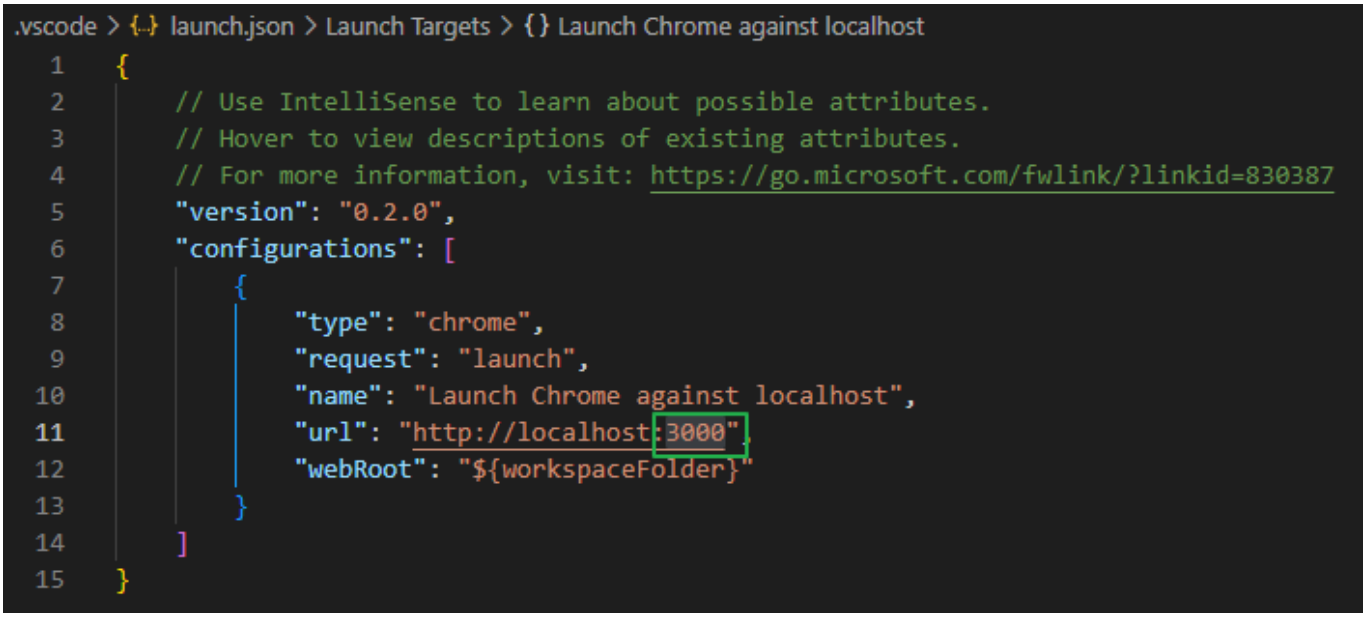
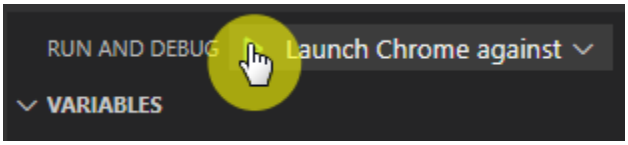
React, Node et MySql Login (Partie 4)

Stockage de l'info de login côté React (Version 1)

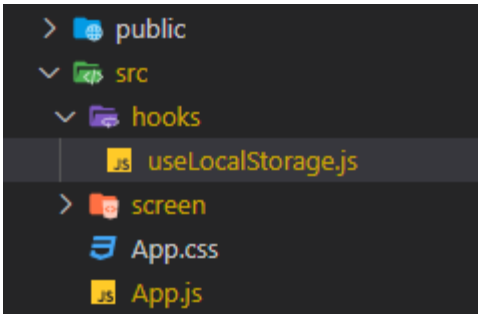
Avant de commencer, sachez qu'il est possible de déboguer du React directement dans VS Code :
1. Ouvrez l'onglet Debug de VS Code et cliquez sur Run and Debug
2. Choisissez l'option Web App (chrome)
3. Modifiez le port dans le fichier launch.json qui vient d'être créé



4. Lancez le serveur (npm start)
5. Mettez des points d'arrêt dans VS Code (idem node)
6. Lancez le Debug (une fenêtre chrome s'ouvre) et le code se met en pause au premier point d'arrêt.



<u>Étape 8 :</u> Dans cette première version, nous allons récupérer les informations de login renvoyées par l'api et les stocker en localStorage.
Nous allons créer un hook personnalisé pour nous aider
Créer le fichier useLocalStorage.js dans un sous dossier hooks de src



Un Hook est un composant qui ne renvoie pas de jsx. Notre hook va renvoyer un state et son setter

src > hooks > useLocalStorage.js > ...

```
1  import { useEffect, useState } from "react";
2
3  const useLocalStorage = (key, defaultValue = null) => {
4
5    const [value, setValue] = useState(
6      JSON.parse(localStorage.getItem(key)) ?? defaultValue
7    );
8
9    useEffect(() => {
10     if (value != null) {
11       localStorage.setItem(key, JSON.stringify(value));
12     } else {
13       localStorage.removeItem(key);
14     }
15   }, [key, value]);
16
17   return [value, setValue];
18 };
19
20 export default useLocalStorage;
21
```

Ln 3 : fonction qui prend en paramètres la clé de stockage en localStorage et une valeur par défaut
Ln 5 à 7 : Création du state initialisé avec la valeur stocké dans le localStorage si elle existe (sinon la valeur qui sera stocké sera la defaultValue)
Ln 9 à 15 : useEffect qui sera exécuté au chargement du composant et à chaque mise à jour de value (valeur stockée en localStorage)
Si la valeur n'est pas null on met à jour le localStorage, sinon on supprime la valeur stockée dans le localStorage
Ln 17 : le hook renvoie le state et son setter

Étape 9 : Modification de LoginScreen

src > screen > LoginScreen.jsx > LoginScreen

```
1  import useLocalStorage from "../hooks/useLocalStorage";
2
3  function LoginScreen(props) {
4    const [auth, setAuth] = useLocalStorage("auth", null);
5
6    const handleSubmit = (e) => {
7      e.preventDefault();
```

Ln 1 : import du hook useLocalStorage
Ln 4 : initialisation du hook avec une clé auth et une valeur null

```
11      const body = JSON.stringify(jsonData);
12      fetch("http://localhost:5000/login", {
13        method: "post",
14        headers: {
15          "content-type": "application/json",
16        },
17        body,
18      })
19        .then((resp) => resp.json())
20        .then((json) => {
21          console.log(json);
22          setAuth(json.data);
23        })
24        .catch((err) => console.log(err));
25    };
```

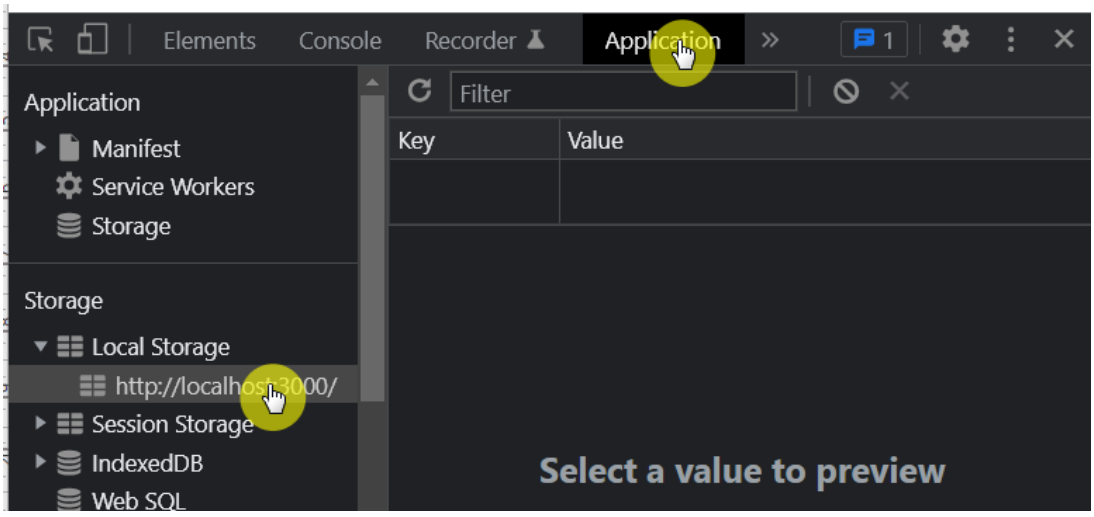
Ln 22 : Une fois la réponse du serveur reçu, nous récupérons les data et utilisons le setter du hook pour mettre à jour la valeur

Fonction logout :

```
25    };
26
27    const handleLogout = (e) => {
28      setAuth(null);
29    };
30
31    return (
32      <>
33        <button className="btn btn-primary" onClick={handleLogout}>
34          Logout
35        </button>
36        <form onSubmit={handleSubmit}>
37          <div className="mb-3">
```

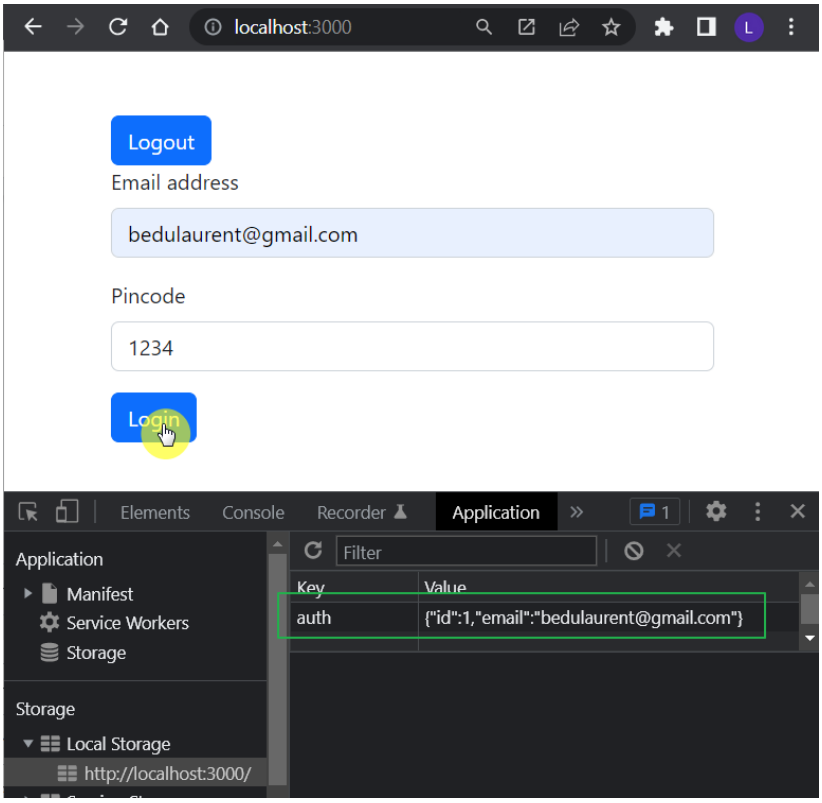
Ln 28 : lors du click sur le bouton Logout (Ln 33 à 35) nous supprimons les infos d'authentification stockées en localStorage

Pour tester, ouvrez l'onglet Application du debugger de Chrome et allez sur Local Storage

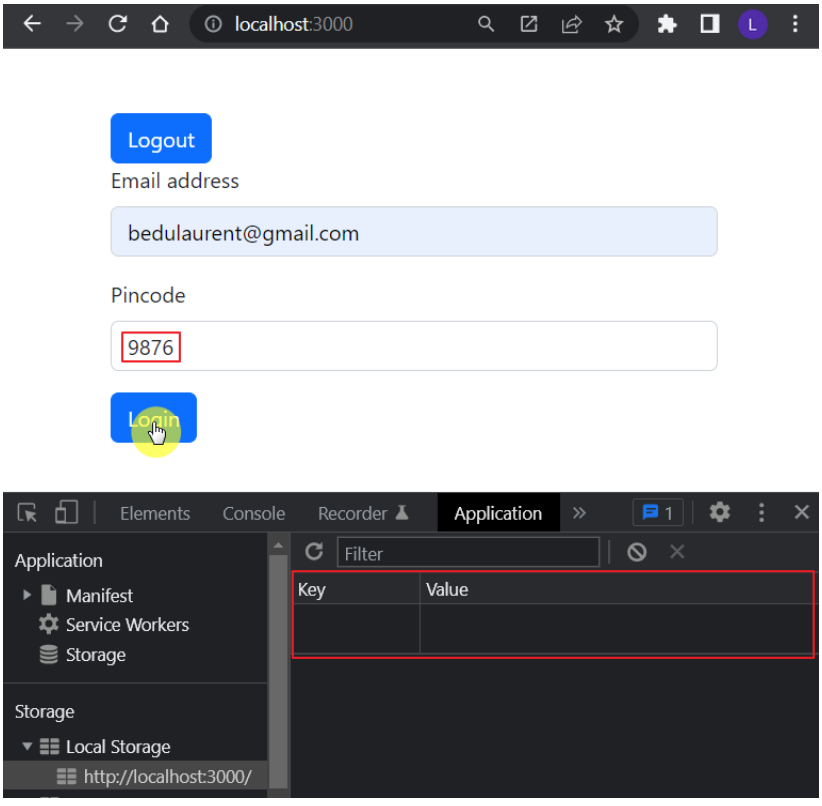


Tester tous les cas et le bouton Logout (suite à un login réussi)

Login OK :



Bad Login :



Logout suite à un Login Ok :