React, Node et MySql Login (Partie 6)

Stockage des infos de login dans un cookie

Une fois connecté côté React (à l'aide du contexte), si vous faites un refresh de la page, l'info de connexion est perdue …
Pour résoudre ce problème, l'API node va devoir renvoyer un token (JWT) qui sera stocké dans un cookie dans le navigateur par l'appli React. C'est beaucoup plus sécurisé que de stocker en clair les infos de login en localStorage

Étape 12 : Création du JWT côté node
Nous allons utiliser le package jsonwebtoken : https://www.npmjs.com/package/jsonwebtoken
Après l'avoir installé le package nous ajoutons une secret key pour le token dans la config

```
3   const config = {
4       dev : {
5           db : {
6               host: "localhost",
7               port: "3306",
8               user: "root",
9               password: "",
10              database: "todo_db"
11          },
12          hash : {
13              prefix: "$2b$08$"
14          },
15          token: {
16              sectet: "Wp].p77qJiLV)jMw6!GXAhzyi(tTaciU"
17          }
18      },
```

…

Puis dans index.js nous ajoutons jwt et renvoyons un token dans la réponse si le login est ok

index.js > ...

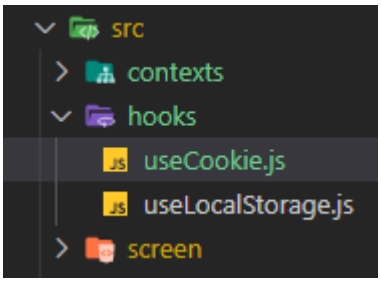
```
const express = require("express");
const app = express();
app.use(express.json());
const { query } = require("./api/services/database.service");
const cors = require("cors");
app.use(cors());
const bcrypt = require("bcrypt");
const config = require("./api/config");
const jwt = require("jsonwebtoken");

app.post("/login", async (req, res) => {
  const { body } = req;
  const sql = `SELECT * FROM customer
               WHERE is_deleted = 0
               AND email = '${body.email}'`;
  await query(sql)
    .then(async (json) => {
      const user = json.length === 1 ? json.pop() : null;
      if (user) {
        const pincodesMatch = await bcrypt.compare(
          body.pincode,
          config.hash.prefix + user.pincode
        );
        if (!pincodesMatch) {
          throw new Error("Bad Login");
        }
        const { id, email } = user;
        const data = { id, email };
        const token = jwt.sign(data, config.token.secret);
        res.json({ data, result: true, message: `Login OK`, token });
      } else {
        throw new Error("Bad Login");
      }
    })
    .catch((err) => {
      res.json({ data: null, result: false, message: err.message });
    });
});
```

Nous pouvons voir le token dans la console suite à un login réussi depuis react

```
                                                    LoginScreen.jsx:24
{data: {…}, result: true, message: 'Login OK', token: 'eyJhbGciOi
▶ JIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZ…5NzF9.FAdzanZnOwx_GiRu7e
tdWFVsw5jvUhAWSTsMSnth4Ek'}
>
```

Étape 13 : Stockage du token dans un cookie

Nous créons un hook pour gérer les cookies



src > hooks > useCookie.js > useCookie > setCookie

```
1   import { useState } from "react";
2
3   const useCookie = (name, defaultValue = null) => {
4
5     const getCookie = (name) => {
6       return (
7         document.cookie.match("(^|;)\\s*" + name + "\\s*=\\s*([^;]+)")?.pop()
8       );
9     };
10
11    const [value, setValue] = useState(getCookie(name) ?? defaultValue);
12
13    const setCookie = (value, options = {}) => {
14      let cookie = name + "=" + value;
15      if (value === null) {
16        cookie = name + "=;max-age=0";
17        document.cookie = cookie;
18        setValue(null);
19        return;
20      }
21      for (let optionKey in options) {
22        cookie += "; " + optionKey;
23        let optionValue = options[optionKey];
24        if (optionValue !== true) {
25          cookie += "=" + optionValue;
26        }
27      }
28      document.cookie = cookie;
29      setValue(cookie);
30    };
31
32    return [value, setCookie];
33  };
34
35  export default useCookie;
```

Mettez des points d'arrêt pour en comprendre le fonctionnement.

Puis nous l'utilisons dans LoginScreen

```
src > screen > LoginScreen.jsx > LoginScreen
1  import { useContext } from "react";
2  import useLocalStorage from "../hooks/useLocalStorage";
3  import { AuthContext } from "../contexts/AuthContext";
4  import useCookie from "../hooks/useCookie";
5
6  function LoginScreen(props) {
7    //const [auth, setAuth] = useLocalStorage("auth", null);
8    const {auth, setAuth} = useContext(AuthContext);
9    const [authCookie, setAuthCookie] = useCookie("auth");
10
11   const handleSubmit = (e) => {
12     e.preventDefault();
13     const form = e.currentTarget;
14     const formData = new FormData(form);
15     const jsonData = Object.fromEntries(formData.entries());
16     const body = JSON.stringify(jsonData);
17     fetch("http://localhost:5000/login", {
18       method: "post",
19       headers: {
20         "content-type": "application/json",
21       },
22       body,
23     })
24       .then((resp) => resp.json())
25       .then((json) => {
26         console.log(json);
27         setAuth(json.data);
28         setAuthCookie(json.token ?? null, {"max-age":`${60*60*24}`});
29       })
30       .catch((err) => console.log(err));
31   };
32
33   const handleLogout = (e) => {
34     setAuth(null);
35     setAuthCookie(null);
36   };
```

Pour en savoir plus sur les cookies :
https://www.pierre-giraud.com/javascript-apprendre-coder-cours/cookies/

Tester un bon login, un bad login et un logout et vérifiez dans l'onglet application du debugger de chrome la présence ou non du cookie.