Création du site dekstop de M2L en react et NodeJS

Accueil.js

front > src > components > Accueil.js > Accueil

```
8   export const Accueil = () => {
9
10      return(
11          <div className='Accueil'>
12              <Carousel bg='dark'>
13                  <Carousel.Item>
14                  <Image
15                      src={imgAccueil2}
16                      width='700px'
17                      height='500px'
18                      rounded
19                      alt='Accueil'
20
21                  />
22                  <Carousel.Caption>
23                      <p>Bienvenue sur M2L !!</p>
24
25                  </Carousel.Caption>
26                  </Carousel.Item>
27                  <Carousel.Item>
28                      <Image
29                          src={imgAccueil}
30                          width='700px'
31                          height='500px'
32                          rounded
33                          alt='Accueil'
34                      />
35                      <Carousel.Caption>
36                          <p>Bienvenue sur M2L !!</p>
37                      </Carousel.Caption>
38                  </Carousel.Item>
39              </Carousel>
40          </div>
41      )
42  }
```

```jsx
<Routes>
    <Route path="/" element={<Accueil />} />
    <Route path="/login" element={<Login account={account} setAccount={setAccount} />} />
    <Route path="/signin" element={<Signin account={account} setAccount={setAccount} />} />
    <Route path="/result" element={<Result userInput={userInput} setUserInput={setUserInput} />} />
    <Route path='/gestionUser' element={<GestionUser />} />
    <Route path='/gestionUser/update/:id' element={<ModifUser />} />
    <Route path='/AddProduct' element={<AddProduct />} />
    <Route path='/gestionProducts' element={<GestionProducts />} />
    <Route path='/gestionProducts/update/:id' element={<ModifProduct />} />
    <Route path='/detailsProduct/:id' element={<DetailsProduct compteurPanier={compteurPanier} setCompteurPanier={setCompteurPanier} />} />
    <Route path='/panier' element={<Panier compteurPanier={compteurPanier} setCompteurPanier={setCompteurPanier} />} />
</Routes>
```

App.js

front > src > components > App.js > App

```jsx
import '../style/App.css';
import { Accueil } from './Accueil';
import { Header } from './Header';
import { Footer } from './Footer';
import { Login } from './Login';
import { Signin } from './Signin';
import { Result } from './Result';
import { Panier } from './Panier';
import { GestionUser } from './GestionUser';
import { GestionProducts } from './GestionProducts';
import { AddProduct } from './CRUD/AddProducts';
import { ModifUser } from './CRUD/ModifUser';
import { ModifProduct } from './CRUD/ModifProduct';
import { DetailsProduct } from './detailsProduct';

import { Routes, Route } from 'react-router-dom';
import { useState } from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';
```

```
  return (
    <>
    <div className='Formulaire'>
    <Form onSubmit={handleSubmit}>
      <Form.Group className="mb-3">
        <Form.Label>Nom d'utilisateur</Form.Label>
        <Form.Control type="text" name='username' placeholder="Nom d'utilisateur" onChange={(e) => setCredentials({...credentials, username: e.target.value})} />
      </Form.Group>

      <Form.Group className="mb-3" controlId="formBasicPassword">
        <Form.Label>Mot de passe</Form.Label>
        <Form.Control type="password" name='password' placeholder="Votre mot de passe" onChange={(e => setCredentials({...credentials, password: e.target.value}))} />
      </Form.Group>
      <Form.Group className="mb-3" controlId="formBasicCheckbox">
        <Form.Check type="checkbox" label="Rester connecté" />
      </Form.Group>
      <Button variant="primary" type="submit">
        Connexion
      </Button>
    </Form>
    </div>
    </>
  );
}
```

Login.js

front > src > components > Login.js > [@] Login > [@] handleSubmit

```
24    const handleSubmit = async (e) => {
25      e.preventDefault();
26      //Crée les données de localStorage
27      localStorage.setItem("username", credentials.username)
28      localStorage.setItem("password", credentials.password)
29      console.log(localStorage.getItem("username"))
30      console.log("Credentials : ", credentials);
31      //Récupère les données égales à ce que l'utilisateur a entré
32      await axios.get(`http://localhost:8000/client/login/${credentials.username}&${credentials.password}`)
33      .then(res => {
34        setClientList(res.data)
35      })
36      console.log(localStorage.getItem("password"))
37      console.log("Client List : ", clientList)
38      //Si la bdd envoie 2 mêmes entrées
39      if(clientList > 2) {
40        alert("Erreur de doublon !")
41      } else {
42        //Si ce que l'utilisateur a entré existe dans la bdd ET si ce compte est admin
43        if(clientList[0].login === localStorage.getItem('username') && clientList[0].mdp === localStorage.getItem('password') && clientList[0].admin === 1) {
44          setAccount({
45            isAuth: true,
46            isAdmin: true
47          })
48          console.log("sensé mettre à jour isAuth : " , account)
49          return alert("Authentification réussie !")
50          //Si ce que l'utilisateur a entré existe dans la bdd ET si ce compte n'est pas admin
51        } else if (clientList[0].login === localStorage.getItem('username') && clientList[0].mdp === localStorage.getItem('password') && clientList[0].admin === 0) {
52          setAccount({
53            isAuth: true,
54            isAdmin: false
55          })
56          return alert("Authentification réussie !")
57
58        } else {
59          return alert("Echec de l'authentification")
60        }
```

GestionUser.js

front > src > components > GestionUser.js > ...

```
export const GestionUser = () => {

    const [research, setResearch] = useState("")
    const [responseAll, setResponseAll] = useState([])


//Récupérer tous les users
    const recupAll = async (event) => {
        await axios.get('http://localhost:8000/client/')
        .then(res => {
            setResponseAll(res.data)
            console.log(res.data)
        })
    }

    const deleteUser = async (id) => {
        console.log(id);
        await axios.delete(`http://localhost:8000/client/delete/${id}`)
        .then(res => {
            console.log(res.data)
        })
        recupAll();
    }

    useEffect(() => {
        recupAll()
    }, [])
```

GestionProducts.js

front > src > components > GestionProducts.js > GestionProducts

```
export const GestionProducts = () => {

    const [responseAll, setResponseAll] = useState([])
    const [research, setResearch] = useState("")



    const recupAll = async () => {
        //Récupère tous les produits
        await axios.get('http://localhost:8000/products/')
        .then(res => {
            setResponseAll(res.data)
            console.log(res.data)
            console.log(responseAll)
        })
    }

    const deleteProduct = async (id) => {
        //Supprime le produit sélectionné
        await axios.delete(`http://localhost:8000/products/delete/${id}`)
        .then(res => {
            console.log(res.data)
        })
        recupAll();
    }

    useEffect(() => {
        recupAll()
    }, [])
```

Installation d’un serveur NodeJS afin de déployer le site dekstop de M2L

Nous ouvrons alors le terminal, puis nous faisons cette commande afin de mettre à jour toutes les sources des dépôts ainsi que de télécharger et installer les mises à jour pour chaque package obsolète

```
root@debian:~# apt update && apt upgrade
```

Puis nous installons Git, npm, NodeJS et mariadb-server

```
root@debian:~# apt install git npm nodejs mariadb-server
```

Et avec cette commande nous sécurisons notre serveur mariadb

```
root@debian:~# mysql_secure_installation
```

Puis nous clonons le projet contenant le site dekstop de M2L, en entrant nos identifiants gitlab

```
root@debian:/opt# git clone https://gitlab.com/souchie88/m2l.git
Clonage dans 'm2l'...
Username for 'https://gitlab.com': souchie88
Password for 'https://souchie88@gitlab.com':
```

Dans la prochaine commande nous importons la base de donnée du site M2L, puis nous créons un utilisateur pour le serveur et nous lui donnons tout les droits sur la base de donnée

```
root@debian:/opt/m2l# sudo mysql -u root -p < db_m2l.sql
```

```
MariaDB [(none)]> CREATE USER 'admin_m2l'@'localhost' IDENTIFIED BY 'admin';
Query OK, 0 rows affected (0,035 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON db_m2l.* TO 'admin_m2l'@'localhost';
Query OK, 0 rows affected (0,008 sec)
```

Nous modifions le .env afin que les informations correspondent avec les informations de la base de donnée et de l'utilisateur

Afin de faire fonctionner le back du site nous installons dans le dossier du back mariadb, cors, express et dotenv

```
npm install mariadb cors express dotenv
```

Puis nous faisons la commande suivante afin d'établir la connexion entre la base de donnée et le serveur NodeJS

```
node Database.js
```

Pour que le site fonctionne il faut créer le fichier serveurnode.service puis mettre ce code dedans

```
root@debian:/opt/m2l/back# touch serveurnode.service /etc/systemd/system
```

```
  GNU nano 5.4                    serveurnode.service
[Unit]
Description=My Node.js App
After=network.target
[Service]
User=sacha
WorkingDirectory=/opt/m2l/back
ExecStart=/usr/bin/node /opt/m2l/back/Database.js
Restart=always
RestartSec=10
[Install]
WantedBy=multi-user.target
```

Afin de savoir sur quel IP le serveur se trouve nous faisons

```
root@debian:/opt/m2l/back# ip a
```

```
inet 10.74.0.247/22
```

Nous activons donc le service daemon puis nous le relançons afin que les modifications soient prise en compte

```
root@debian:/etc/systemd/system# systemctl enable serveurnode.service
```

```
root@debian:/etc/systemd/system# systemctl daemon-reload
```

Le site est déployé, nous entrons l'IP dans le navigateur en précisant quelle route nous voulons tester, voici la route client