Création d'une application MERN – Partie 1 : Front end

Cette partie est la suite du TP précédent. L'objectif est de créer une application frontend en utilisant React. Comme vous avez pu le constater, react se base principalement sur des composants. Chaque composant est sous forme d'une fonction qui retourne du code sous forme de balises. Ce code ressemble au code HTML, il s'agit en réalité des composants react qui ressemblent beaucoup aux balises HTML.

Dans cette partie, vous allez découvrir les routes, comment passer des paramètres entres les différentes pages de votre application

1. Ajouter des routes

- Ouvrir l'application E-Commerce sur laquelle vous avez travaillé pour ajouter des routes.
- Depuis le terminal, vérifier que vous êtes sur le dossier frontend ou bien y accéder depuis le terminal.
- Lancer la commande suivante pour installer le module router : npm i react-router-dom comme le montre la figure suivante :

```
PS C:\Users\ACER\Desktop\EcommerceApp\frontend> npm i react-router-dom
```

- Installer également le module react-route-boostrap comme le montre la figure suivante

```
PS C:\Users\ACER\Desktop\EcommerceApp\frontend> npm i react-route-bootstrap
[...................] - idealTree: sill logfile start cleaning logs, removing 2 f
```

- Ouvrir le fichier App.js et importer les éléments Route, Router, Routes comme le montre la figure suivante :

```
import { BrowserRouter as Router, Route, Routes} from 'react-router-dom'
```

- Ensuite, changer le contenu comme suit:

```
<>
  <Router>
   <Header/>
   <Routes>
    <Route exact path="/"  element={<ProductList /> }></Route>
   </Routes>
   <Footer/>
  </Router>
</>
```

- Tester l'application pour voir si les routes que venez d'ajouter fonctionnent correctement.

- Ajouter une nouvelle route permettant d'afficher les détails d'un produit lorsque l'utilisateur appuie sur le bouton «Add to card». Comme vous pouvez le remarquer, cette route prend l'id du produit comme paramètre qui sera passé à la page ProductDetails en vue d'afficher plus de détails sur e produit

```
<Route path='/product/:id' element={<ProductDetails/>}></Route>
```

- Toujours dans le dossier component, créer un nouveau fichier javascript qui permettra d'afficher les détails d'un produit que vous appellerez ProductDetails.js
- Ouvrir le fichier ProductDetail.js et y ajouter le contenu suivant avec le raccourci rafce

```
import React from 'react'

const ProductDetails = () => {
  return (
    <div> ProductDetails </div>
  )
}
export default ProductDetails
```

- Remplacer les avec des balises <></> qui vont encapsuler tout le code de l'application :
- A l'intérieur, ajouter un lien permettant de revenir toujours à la page d'accueil

```
<Link className='btn btn-light my-3' to='/'>
        Go Back
      </Link>
```

- N'oubliez pas d'inclure la bibliothèque :

```
import { Link } from 'react-router-dom'
```

2. Passage de paramètres entre les composants react

Maintenant vous avez construit deux pages de votre application. La première affiche le catalogue des produits, et la deuxième affiche les détails du produit qui a été sélectionné par l'utilisateur. Dans ce sens, le composant appelé ProductList doit envoyer l'id du produit selectionné au composant appelé ProductDetails.

- Ouvrir le script ProductDetails.js
- Importer useParams de react-router-dom

```
import { useParams } from 'react-router-dom';
```

- A l'intérieur de la fonction *ProductDetails* ajouter le code suivant :

```
const ProductDetails = () => {
    const {id} = useParams();
    const product = products.find((p) => p._id === id)
  return (
    <>
```

- La première ligne permet de récupérer le paramètre appelé id. Du moment que la méthode useParams() retourne tous les paramètres, il est important d'utiliser les {} pour spécifier quel paramètre vous souhaitez récupérer.
- Dans la deuxième ligne, nous déclarons produit appelé «product » auquel nous allons affecter un produit récupéré depuis le fichier JSON products.js. Pour chercher un produit dans le fichier JSON, il suffit d'utiliser la méthode find() qui permet dans ce cas de chercher un produit (p) dont l'id est celui récupéré depuis les paramètres dans l'instruction précédente.
- A l'intérieur de la fonction return() ajouter le code suivant après la balise <Link>

```
<Row>
            <Col>
              <Image src={'../'+product.image} alt={product.name} fluid />
            </Col>
</Row>
```

- Exécuter pour vérifier le résultat
- Ensuite, à l'intérieur de la balise <Row></Row>, ajouter la deuxième colonne, affichant le nom et la description du produit comme suite :

```
 <Col>
              <ListGroup variant='flush'>
                <ListGroup.Item>
                  <h3>{product.name}</h3>
                </ListGroup.Item>

                <ListGroup.Item>
                  Description: {product.description}
                </ListGroup.Item>
              </ListGroup>
 </Col>
```

Finalement, Ajouter une troisième colonne pour afficher le prix et vérifier si le produit sélectionné est en stock

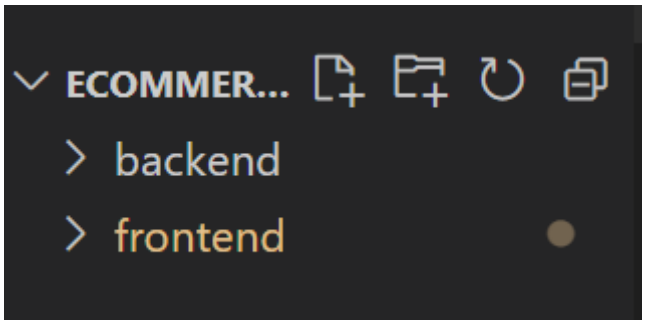
```
<Col>
            <ListGroup>
                  <ListGroup.Item>
                    <Row>
                      <Col>Price:</Col>
                      <Col>
                        <strong>${product.price}</strong>
                      </Col>
                    </Row>
                  </ListGroup.Item>

                  <ListGroup.Item>
                    <Row>
                      <Col>Status:</Col>
                      <Col>
                        {product.countInStock > 0 ? 'In Stock' : 'Out Of Stock'}
                      </Col>
```

```
                    </Row>
                  </ListGroup.Item>
                </ListGroup>
              </Col>
```

3. Construction du backend

Dans la racine du projet, créer un nouveau dossier que vous appellerez «backend». Le backend sera sous forme d'une application Node.js qui utilise le module express pour démarrer un serveur web sur le port 5000. L'objectif du serveur est de recevoir les requêtes https et renvoyer les données des produits demandés dans la requête.

Le contenu de votre application doit ressembler à ceci:



- Depuis le terminal, accéder au dossier backend et intiliser un projet Node.js avec la commande npm init.

```
PS C:\Users\ACER\Desktop\EcommerceApp>   cd backend
PS C:\Users\ACER\Desktop\EcommerceApp\backend>  npm init --yes
Wrote to C:\Users\ACER\Desktop\EcommerceApp\backend\package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

- Installer le mode express ainsi que le module nodemon
- Modifier le contenu du fichier package.json du dossier backend comme suit :

backend > {} package.json > {} scripts

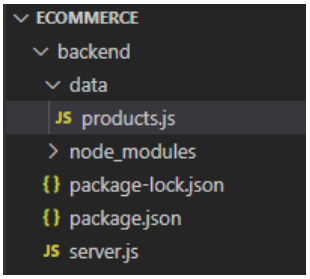
```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  Debug
  "scripts": {
    "start": "nodemon server.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
```

- Dans le dossier backend, créer un fichier appelé server.js. Y copier le contenu ci-dessous et l'exécuter grâce à la commande npm start

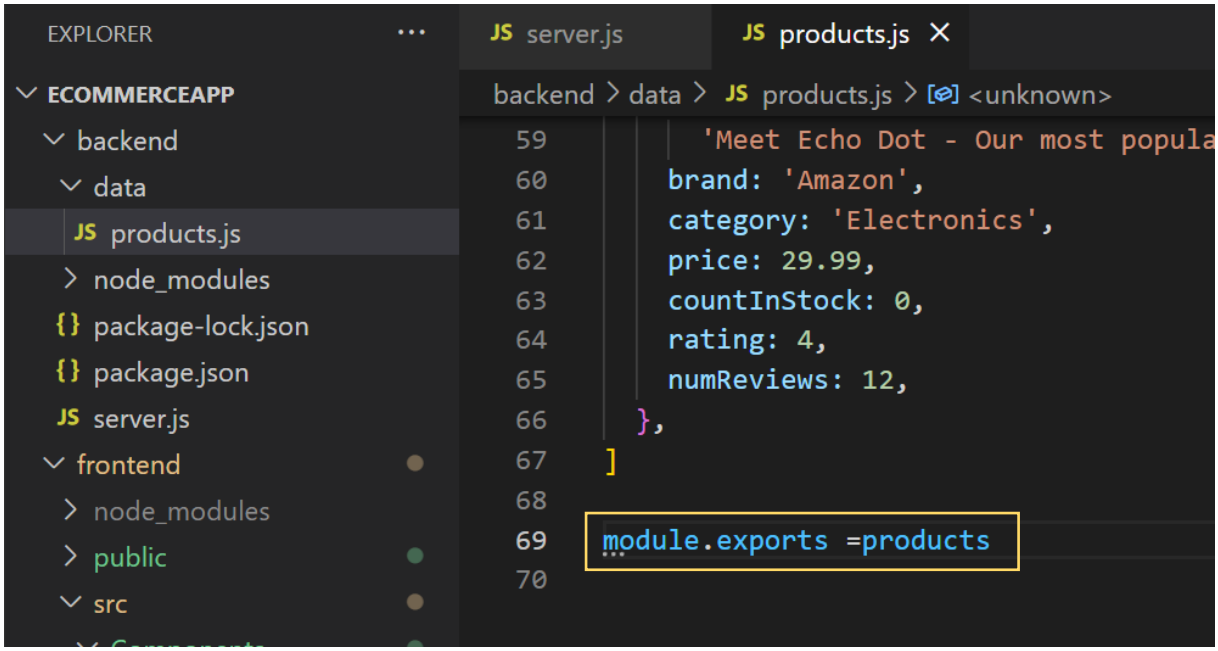
```
const express= require('express')
const app = express()

app.get('/',(req,res)=>{

    res.send("Testing API")
})

app.listen(5000,()=>{
    console.log("Server started and running on port 5000")
})
```

A ce stade, vous avez deux applications. Une application frontend avec react qui fonctionne sur le port 3000 et une application backend avec node.js et express qui fonctionne sur le port 5000.

Dans le dossier Backend, créer un nouveau dossier que vous appellerez Data. Et y copier le fichier appelé product.js (le fichier JSON contenant des exemples de produits=



Modifier le contenu du fichier de données comme le montre la figure suivante:



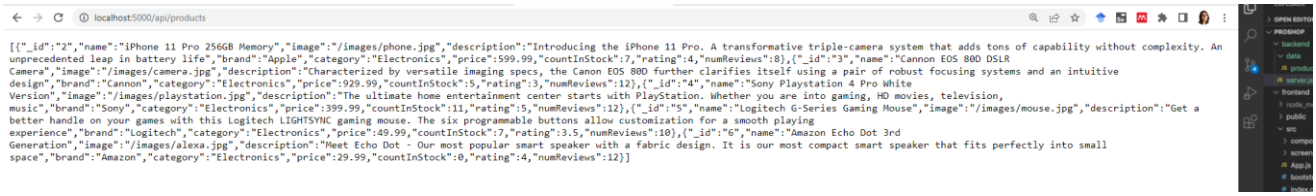
- Modifier le contenu du fichier server.js pour inclure le fichier des produits

```
const products= require('./data/products')
```

- Dans le fichier server.js, ajouter une nouvelle route permettant de récupérer tous les produits :

```
app.get('/api/products', (req,res)=>{
res.json(products)
}
```

Tester la route, vous devriez avoir un résultat qui ressemble à ceci.



- Ajouter une nouvelle route dans le fichier server.js qui permet de retourner un produit spécifique dans l'id est un paramètre de requête http

```
app.get('/api/products/:id', (req,res)=>{
  const product= products.find((p)=> p._id===req.params.id)
  res.json(product)
  }
  )
```

- Tester la route pour s'assurer qu'elle fonctionne

4. Recevoir les données depuis le backend et les afficher dans le frontend
    - Accéder au dossier frontend et installer le module axios
    - Accéder au backend et installer le module cors npm install cors –save
    - Ouvrir le script ProductDetails.js et y ajouter le contenu suivant :

```
import React, { useState, useEffect }  from 'react'
import axios from "axios"
```

    - Vous pouvez recupérer la liste des produits en utilisant le code suivant :

```
const ProductDetails = () => {
  const[product, setProducts]=useState([])
  useEffect(()=>
  {
    const fetchProducts= async ()=>{
const {data}= await axios.get(('/api/products/'))
setProducts(data)
    }
    fetchProducts()
  },[])
      return (
```

La fonction "useState" s'utilise comme tous les Hooks au sein d'un composant fonctionnel. Elle prend en paramètre l'état initial de votre composant dans notre cas un tableau vide à remplir avec les produits et retourne un tuple composé de 2 éléments : le tableau des produits et la fonction qui va mettre à jour cet état.

- useEffect, un hook qui va permettre de déclencher une fonction de manière asynchrone lorsque l'état du composant change. Dans notre cas la fonction useEffect() nous permettra de déclencher le fetch au chargement de la page
- useState permettra de stocker le retour de l'API dans le state .

-Avant de lancer votre projet, assurer vous d'ajouter le module cors à votre backend (Server.js) comme suit:

```
var cors = require('cors')
app.use(cors())
```

- Tester la backend et le frontend de votre application.