Tipo : Guía de laboratorio
Capítulo : React JS
Duración : 60 minutos

I. OBJETIVO

Crear un proyecto React que consuma datos desde servicios externos.

II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Editor Visual Studio Code
- NodeJS

III. EJECUCIÓN DEL LABORATORIO

- **Ejercicio 2.5: Consumir datos desde servicios externos**

1. Instalar la librería **axios** para hacer peticiones http.

```
npm install --save axios
```

2. Crear el archivo **db.json** con el schema productos.

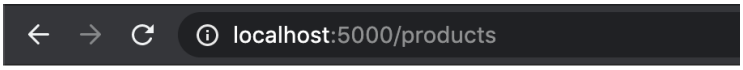
{} db.json

mocks > {} db.json > ...

```
{
    "products": [
        {
            "id": 2,
            "name": "Televisor 49\" 4K UHD",
            "detail": "Modelo: 49UK6200",
            "price": "1499.00",
            "stock": 8
        },
        {
            "id": 3,
            "name": "Laptop 15\" AMD A6 4GB 500GB",
            "detail": "Módelo: IdeaPad 330",
            "price": "1299.00",
            "stock": 0
        }
    ]
}
```

3. Correr el REST api falso

```
json-server db.json --watch --port=5000
```




```
[
  {
    "id": 2,
    "name": "Televisor 49\" 4K UHD",
    "detail": "Modelo: 49UK6200",
    "price": "1499.00",
    "stock": 8
  },
  {
    "id": 3,
    "name": "Laptop 15\" AMD A6 4GB 500GB",
    "detail": "Módelo: IdeaPad 330",
    "price": "1299.00",
    "stock": 0
  }
]
```


4. Crear un Servicio para consumir el REST API implementado

productService.js

src > services > productService.js > ...

```
import axios from 'axios';

export const ProductService = {
    API: 'products',
    getProducts() {
        const URL = `${process.env.REACT_APP_API_URL}/${this.API}`;
        return axios.get(URL);
    },
    getProduct(id) {
        const URL = `${process.env.REACT_APP_API_URL}/${this.API}/${id}`;
        return axios.get(URL);
    },
    removeProduct(id) {
        const URL = `${process.env.REACT_APP_API_URL}/${this.API}/${id}`;
        return axios.delete(URL);
    },
    createProduct(product) {
        const URL = `${process.env.REACT_APP_API_URL}/${this.API}`;
        return axios.post(URL, product);
    },
    updateProduct(product) {
        const URL = `${process.env.REACT_APP_API_URL}/${this.API}/${product.id}`;
        return axios.put(URL, product);
    }
}
```

5. Actualizar el componente **Products** para hacer uso de este nuevo servicio implementado.

```
componentDidMount() {
    ProductService.getProducts()
        .then(response => {
            const { data: products } = response;
            this.setState({ products, filteredProducts: products });
        });
}

handleDelete = id => {
    ProductService.removeProduct(id)
        .then(() => {
            const products =  this.state.products.filter(product => product.id !== id);

            this.setState({
                products,
                filteredProducts: products
            });
        });
};
```

6. Actualizar la configuración del router en el **App** componente para obtener la referencia de la navegación en los componentes.

App.js

```
src > App.js > ...
1   import React from 'react';
2   import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
3
4   import Footer from './modules/admin/Footer';
5   import Header from './modules/admin/Header';
6   import Products from './modules/admin/Products';
7   import NewProduct from './modules/admin/NewProduct';
8   import EditProduct from './modules/admin/EditProduct';
9
10  function App() {
11    return (
12      <div className="App">
13        <Header />
14          <Router>
15            <Switch>
16              <Route exact path='/admin/products' component={Products}/>
17              <Route path='/admin/products/new' component={NewProduct} />
18              <Route path='/admin/products/:id' component={EditProduct} />
19            </Switch>
20          </Router>
21        <Footer />
22      </div>
23    );
24  }
25
26  export default App;
```

7. Obtener el producto a editar en el componente **EditProduct**.

```
import { ProductService } from '../../services/productService';

class EditProduct extends Component {
    state = {
        product: {}
    }

    async componentDidMount() {
        const { params: { id } } = this.props.match;
        const { data: product } = await ProductService.getProduct(id);
        this.setState({ product });
    }

    render() {
        const { product } = this.state;

        return (
            <div className="columns">
                <div className="column is-10 is-offset-1">
                    <h2 className="subtitle is-3 has-text-centered">
                        Actualizar Producto: [{ product.id }]
                    </h2>
                </div>
            </div>
        );
    }
}
```

IV. EVALUACIÓN

1. ¿Qué son los verbos http?

   HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Estos métodos de solicitud a veces son llamados HTTP verbs. Los verbos http son: GET, POST, PUT, DELETE, PATCH, OPTIONS, entre otros más.

2. ¿Qué es axios?

   Axios es un cliente HTTP basado en Promesas para Javascript, el cual puede ser utilizado tanto en tu aplicación Front-end.