

Tipo : Guía de laboratorio
Capítulo : React JS
Duración : 45 minutos

I. OBJETIVO

Crear un proyecto React donde los componentes puedan comunicarse e intercambiar información.

II. REQUISITOS

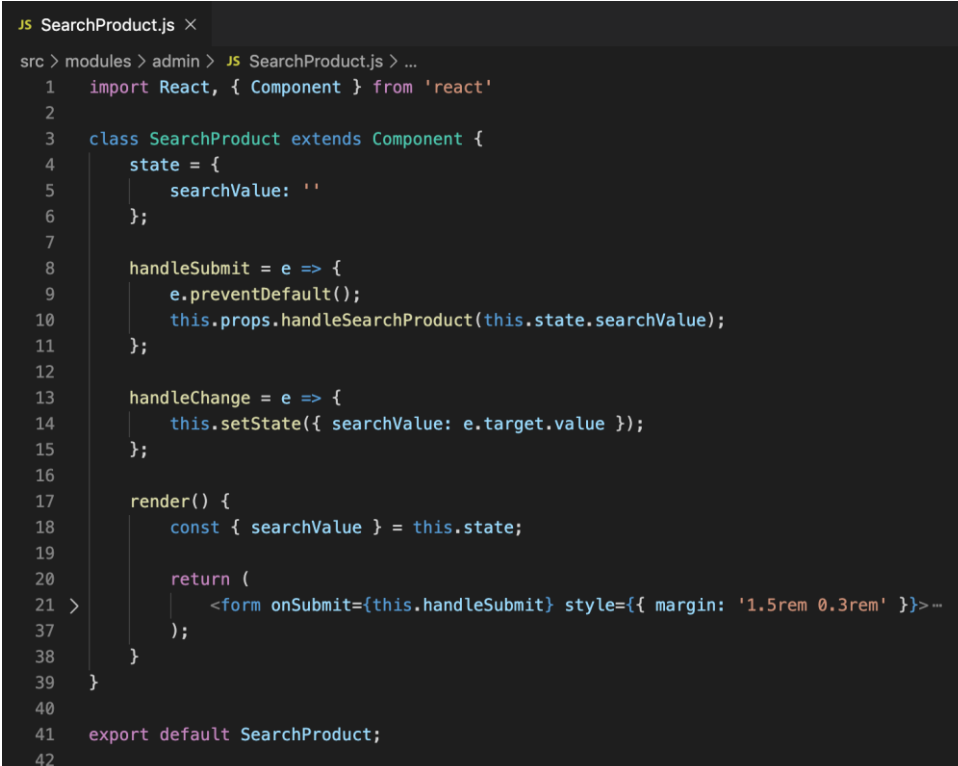
Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Editor Visual Studio Code
- NodeJS

III. EJECUCIÓN DEL LABORATORIO

- **Ejercicio 2.3: Comunicar la información entre componentes**

1. Crear el componente **SearchProduct**.



```
JS SearchProduct.js ×
src > modules > admin > JS SearchProduct.js > ...
1  import React, { Component } from 'react'
2
3  class SearchProduct extends Component {
4    state = {
5      searchValue: ''
6    };
7
8    handleSubmit = e => {
9      e.preventDefault();
10     this.props.handleSearchProduct(this.state.searchValue);
11   };
12
13   handleChange = e => {
14     this.setState({ searchValue: e.target.value });
15   };
16
17   render() {
18     const { searchValue } = this.state;
19
20     return (
21 >     <form onSubmit={this.handleSubmit} style={{ margin: '1.5rem 0.3rem' }}>--
37     );
38   }
39 }
40
41 export default SearchProduct;
42
```

2. Actualizar el estado del componente **Products**.

```

6   const products = [
7     {
8       id: 2,
9       name: 'Televisor 49" 4K UHD',
10      detail: 'Modelo: 49UK6200',
11      price: '1499.00',
12      stock: 8,
13    },
14    {
15      id: 3,
16      name: 'Laptop 15" AMD A6 4GB 500GB',
17      detail: 'Modelo: IdeaPad 330',
18      price: '1299.00',
19      stock: 0,
20    }
21  ];
22
23  class Products extends Component {
24    constructor(props) {
25      super(props);
26      this.state = {
27        titles: ['#', 'Nombre', 'Detalle', 'Precio', 'Stock', 'Acciones'],
28        filteredProducts: products,
29        products: products
30      };
31    }
32  }

```

3. Crear el método que realizará el filtrado de productos.

src/modules/admin/Products.js

```

handleSearchProduct = searchValue => {
  if (searchValue) {
    const value = searchValue.toLowerCase();
    this.setState(prevState => {
      return {
        filteredProducts: prevState.products.filter(product => (
          product.name.toLowerCase().includes(value)
        ))
      };
    });
  } else {
    this.setState(prevState => {
      return {
        filteredProducts: prevState.products
      };
    });
  }
};

```

4. Pasarle por **props** al componente **SearchProduct** el método.

src/modules/admin/Products.js

```

</div>
<SearchProduct handleSearchProduct={this.handleSearchProduct}/>
<table className="table is-striped is-hoverable is-bordered is-fullwidth">
  <thead>

```

5. Llamar al método **handleSerachProduct** del componente padre y pasarle el valor de búsqueda.

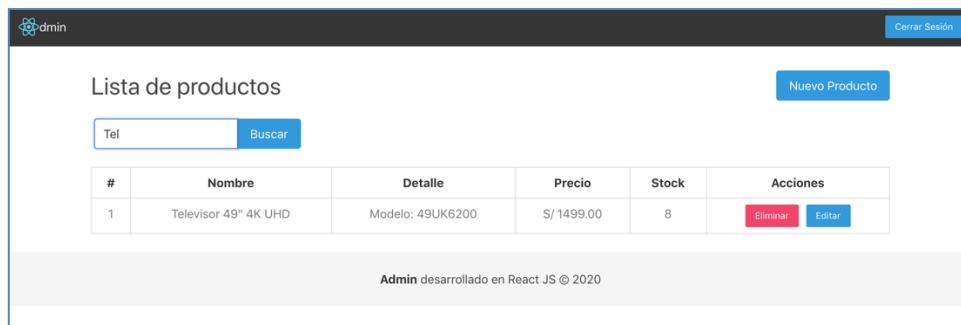
src/modules/admin/SearchProduct.js

```
handleSubmit = e => {
  e.preventDefault();
  this.props.handleSearchProduct(this.state.searchValue);
};

handleChange = e => {
  this.setState({ searchValue: e.target.value });
};

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

  return (
    <form onSubmit={this.handleSubmit} style={{ margin: '1.5rem 0.3rem' }}>
      <div className="field has-addons">
        <div className="control">
          <input
            className="input"
            type="text"
            value={searchValue}
            placeholder="Buscar productos"
            onChange={this.handleChange}
          />
        </div>
      </div>
    </form>
  );
}
```



The screenshot shows a web application interface for an admin panel. At the top, there's a header with a logo and the text 'admin', and a 'Cerrar Sesión' button. The main content area is titled 'Lista de productos' and includes a 'Nuevo Producto' button. Below the title, there's a search bar with a 'Tel' input field and a 'Buscar' button. A table displays a list of products with columns for '#', 'Nombre', 'Detalle', 'Precio', 'Stock', and 'Acciones'. The first row shows a product with ID 1, named 'Televisor 49" 4K UHD', with details 'Modelo: 49UK6200', price 'S/ 1499.00', and stock '8'. The 'Acciones' column for this product contains 'Eliminar' and 'Editar' buttons. At the bottom, a footer states 'Admin desarrollado en React JS © 2020'.

#	Nombre	Detalle	Precio	Stock	Acciones
1	Televisor 49" 4K UHD	Modelo: 49UK6200	S/ 1499.00	8	<button>Eliminar</button> <button>Editar</button>

IV. EVALUACIÓN

1. ¿Cuál es la diferencia entre **props** y **state** en React?

Los props es lo que recibe un componente al ser renderizado, son como las propiedades de un elemento html. En este caso pueden tener propiedades de cualquier tipo, y si son funciones, estas sirven para comunicarse con el componente padre.

El state de los componentes sirve para manejar datos de manera local y poder responder a sus cambios. A diferencia de los props estos pueden cambiar su valor durante la existencia del component, mientras que los props son solo de lectura.

2. ¿Por qué razón debemos pasar asignar una propiedad key cuando renderizamos una lista de elementos en React?

Las keys ayudan a React a identificar que items han cambiado, son agregados, o son eliminados.