

CONSUMIR API REST EN REACT CON AXIOS.

Vamos a aprender cómo consumir servicios en React con axios de una API externa (Api de Clientes), vamos a realizar un ejemplo práctico paso a paso. Mediante este ejemplo accederemos a la api para poder realizar un CRUD (Create, Read, Update, Delete) de Clientes con React. Además, en este caso usaremos el componente axios que nos proporciona React.

Vamos a usar navegación entre componentes y clases de bootstrap para tener un diseño más claro, aunque sencillo.

1. Crear maqueta del Proyecto recat React UI con "Create React App".
2. Agregar Bootstrap y Axios en React usando NPM.
3. Componente de Navegación MenuClientes.
4. Componente Clientes.
5. Componente InsertarClientes.
6. Componente DetallesCliente
7. Componente UpdateCliente
8. Componente DeleteCliente

1. Crear React UI con "Create React App"

En el ejercicio al creamos la maqueta del proyecto con la ayuda de la aplicación llamada "Create React App" que es ideal para crear una aplicación de una sola página. Puedes revisar la documentación oficial en el sitio: <https://create-react-app.dev/>.

`npx create-react-app frontclientes`

```
npx create-react-app my-app
cd my-app
npm start
```

2. Agregar Dependencias en React usando NPM.

Para realizar las peticiones utilizaremos AXIOS, debido a que es compatible con la mayoría de navegadores, incluso con los más antiguos, para la parte de los estilos utilizaremos Bootstrap.

- npm i axios

Axios.

Axios es un cliente HTTP ligero que se basa en promesas. Es un componente de React que permite interactuar con una API REST. Por eso lo instalamos en nuestro proyecto de react.

```
λ npm install --save react-router-dom
```

```
λ npm install bootstrap --save
```

```
λ npm install jquery popper.js
```

React Router.

React Router es una colección de componentes de navegación la cual podemos usar como ya lo mencione tanto en web o en móvil con React Native. Con esta librería vamos a obtener un enrutamiento dinámico gracias a los componentes, en otras palabras tenemos unas rutas que renderizan un componente.

3. Ajuste de App.

Sobre la carpeta de App se mueve el archivo App.js y se agrega el siguiente contenido.

```
src > components > App > JS App.js > ...
1  import './App.css';
2  import "bootstrap/dist/css/bootstrap.min.css";
3  import "bootstrap/dist/js/bootstrap.bundle.min";
4  import Router from '../Router';
5
6
7  function App() {
8    return (
9      <div className="App">
10       <Router />
11     </div>
12   );
13 }
14
15 export default App;
16
```

- **Clase Router**

Agregamos la clase Router que permite navegar entre los componentes a utilizar en el proyecto.

```
src > components > JS Router.js > Router > render
1  import React, { Component } from 'react';
2  import { BrowserRouter, Route, Switch } from 'react-router-dom';
3  import MenuClientes from './CrudClientes/MenuClientes';
4  import Clientes from './CrudClientes/Clientes';
5  import DetallesCliente from './CrudClientes/DetallesCliente';
6  import UpdateCliente from './CrudClientes/UpdateCliente';
7  import DeleteCliente from './CrudClientes/DeleteCliente';
8  import InsertarCliente from './CrudClientes/InsertarCliente';
9
```

```

10 export default class Router extends Component {
11   render() {
12     return (
13       <div>
14         <BrowserRouter>
15           <MenuClientes />   Estático
16           <Switch>
17             <Route exact path="/" component={Clientes} />
18             <Route exact path="/create" component={InsertarCliente} />
19             <Route exact path="/detalles/:_id" render={props => {
20               var id = props.match.params._id;
21               return <DetallesCliente _id={id} />
22             }} />
23             <Route exact path="/update/:_id" render={props => {
24               var id = props.match.params._id;
25               return <UpdateCliente _id={id} />
26             }} />
27             <Route exact path="/delete/:_id" render={props => {
28               var id = props.match.params._id;
29               return <DeleteCliente _id={id} />
30             }} />
31           </Switch>
32         </BrowserRouter>
33       </div>
34     )
35   }
36 }

```

Dinámico

• Componente de Navegación MenuClientes.

```

1  import React, { Component } from 'react';
2  import { NavLink } from 'react-router-dom';
3
4  export default class MenuClientes extends Component {
5    render() {
6      return (
7        <nav className="navbar navbar-expand-lg navbar-dark bg-primary">
8          <a className="navbar-brand" href="/">Navbar</a>
9          <button className="navbar-toggler" type="button" data-toggle="collapse"
10             data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown"
11             aria-expanded="false" aria-label="Toggle navigation">
12             <span className="navbar-toggler-icon"></span>
13           </button>
14           <div className="collapse navbar-collapse" id="navbarNavDropdown">
15             <ul className="navbar-nav">
16               <li className="nav-item active">
17                 <NavLink className="nav-link" to="/">Clientes <span className="sr-only">
18                   (current)</span></NavLink>
19               </li>
20               <li className="nav-item">
21                 <NavLink className="nav-link" to="/create">Nuevo Cliente</NavLink>
22               </li>
23             </ul>
24           </div>
25         </nav>
26       )
27     }
28   }

```

4. Componente Clientes.

Antes de crear las clases para las distintas peticiones creamos la clase Global, en la que tendré las urls de mis APIs (en este caso solo accederemos a una) y que importaré en todas las clases que la necesiten.

```
src > JS Global.js > [⌘] default
1   var Global = {
2     |   urlclientes: "http://localhost:8082/api/clientes"
3   }
4
5   export default Global;
```

Este componente Clientes va a contener una tabla con todos los clientes disponibles dentro de una carpeta llamada CrudClientes.

La forma de crear este componente será similar en los siguientes, variando en pequeños detalles debido a las distintas peticiones. Por eso se explicará a fondo este y los demás solo se comentarán aportando contenido visual.

Este componente necesita importar axios, ya que va a realizar una petición a un servicio, en este caso GET para recibir clientes, .

```
1   import React, { Component } from 'react';
2   import axios from 'axios';
3   import Global from '../Global';
4   import { NavLink } from 'react-router-dom';
5
```

En la clase Clientes vamos a inicializar dos variables de estado para que se actualice la vista dependiendo de su valor. Tendremos un array vacío 'clientes' para guardar los clientes y otra 'status' a 'false' para el estado.

```
6   export default class Clientes extends Component {
7
8     state = {
9       clientes: []
10      , status: false
11    }
12
13    cargarClientes = () => {
14      var url = Global.urlclientes;
15      var request = "/clientes";
16      axios.get(url + request).then(res => {
17        this.setState({
18          clientes: res.data
19          , status: true
20        });
21      });
22    }
23
24    componentDidMount = () => {
25      this.cargarClientes();
26    }
27  }
```

Creamos la función `componentDidMount` (métodos de ciclos de vida) en la que llamaremos al método `cargarClientes`, que va a recibir todos los clientes de la API. Además usaremos la url que tenemos en `Global` y le añadiremos la ruta específica que queremos, en este caso `'/clientes'`.

Usamos el método `'get'` de `axios` al que le enviamos como parámetro el string de la url completa (`Global + petición`). En la promesa que hace `axios` con `'then'` (una vez que haya conectado correctamente) obtiene una respuesta `'res'`.

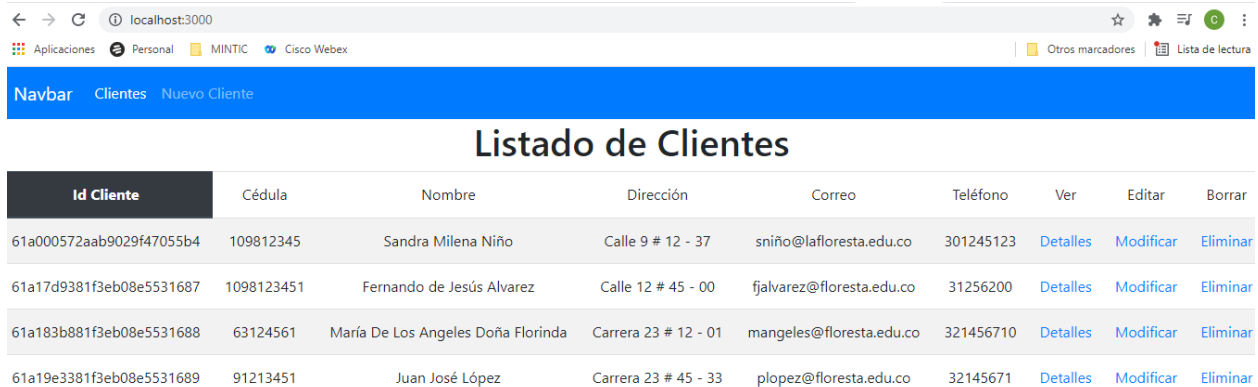
Ahora modificamos nuestras variables de estado para guardar los datos obtenidos: guardamos en `'clientes'` los datos de la respuesta con `'res.data'` y `'status'` lo ponemos a `'true'` para usarlo como condición a la hora de pintar nuestros datos en la vista.

Solo falta el `'render'`, que será la parte visual del componente `Clientes`.

En él dibujamos un título y una tabla. En el cuerpo de la tabla, como queremos que sea dinámico dependiendo de los clientes que obtenga en la petición, vamos a usar la variable de estado 'status' como condición para mostrar los datos. Una vez que esta variable es 'true' (cuando se modifica al hacer la petición correctamente con axios) recorreremos el array 'clientes' con '.map' y por cada línea de la tabla mostramos los datos de cada cliente.

```
28     render() {
29         return (
30             <div>
31                 <h1>Listado de Clientes</h1>
32                 <table className="table table-striped">
33                     <thead className="thead-dark">
34                         <tr>
35                             <th>Id Cliente</th>
36                             <td> Cédula</td>
37                             <td> Nombre</td>
38                             <td> Dirección</td>
39                             <td> Correo</td>
40                             <td> Teléfono</td>
41                             <td> Ver</td>
42                             <td> Editar</td>
43                             <td> Borrar</td>
44                         </tr>
45                     </thead>
46                     <tbody>
47                         {this.state.status === true &&
48                         (
49                             this.state.clientes.map(cliente => {
50                                 return(
51                                     <tr key={cliente._id}>
52                                         <td> {cliente._id}</td>
53                                         <td> {cliente.cedula}</td>
54                                         <td> {cliente.nombre}</td>
55                                         <td> {cliente.direccion}</td>
56                                         <td> {cliente.email}</td>
57                                         <td> {cliente.telefono}</td>
58                                         <td>
59                                             <NavLink to={"/detalles/" + cliente._id}>Detalles</NavLink>
60                                         </td>
61                                         <td>
62                                             <NavLink to={"/update/" + cliente._id}>Modificar</NavLink>
63                                         </td>
64                                         <td>
65                                             <NavLink to={"/delete/" + cliente._id}>Eliminar</NavLink>
66                                         </td>
67                                     </tr>
68                                 );
69                             })
70                         )
71                     </tbody>
72                 </table>
73             </div>
74         )
75     }
76 }
77 }
```

Para probar el proyecto debemos levantar el Api Rest construido en Spring Boot y después ejecutamos nuestro frontend con **npm start**, el resultado se puede apreciar en la figura.



Id Cliente	Cédula	Nombre	Dirección	Correo	Teléfono	Ver	Editar	Borrar
61a000572aab9029f47055b4	109812345	Sandra Milena Niño	Calle 9 # 12 - 37	sniño@lafloresta.edu.co	301245123	Detalles	Modificar	Eliminar
61a17d9381f3eb08e5531687	1098123451	Fernando de Jesús Alvarez	Calle 12 # 45 - 00	fjalvarez@floresta.edu.co	31256200	Detalles	Modificar	Eliminar
61a183b881f3eb08e5531688	63124561	María De Los Angeles Doña Florinda	Carrera 23 # 12 - 01	mangeles@floresta.edu.co	321456710	Detalles	Modificar	Eliminar
61a19e3381f3eb08e5531689	91213451	Juan José López	Carrera 23 # 45 - 33	plopez@floresta.edu.co	32145671	Detalles	Modificar	Eliminar

5. Componente InsertarClientes

Para crear este componente necesitamos importar también axios y Global, y para insertar un cliente nuevo debemos tener sus datos. En nuestro caso los obtenemos mediante un formulario.

```
src > components > CrudClientes > JS InsertarCliente.js > InsertarCliente > cajaTelRef
1  import React, { Component } from 'react';
2  import axios from 'axios';
3  import Global from '../Global';
4  import { Redirect } from 'react-router-dom';
5
6  export default class InsertarCliente extends Component {
7
8      cajaCedRef = React.createRef();
9      cajaNomRef = React.createRef();
10     cajaDirRef = React.createRef();
11     cajaEmaRef = React.createRef();
12     cajaTelRef = React.createRef();
13
14     state = { status: false }
15 }
```

Creamos un método nuevaCliente en el que guardaremos los datos

obtenidos del nuevo cliente en variables para asignarlas a las propiedades de un objeto json 'clientes'. También guardamos la url completa para nuestra petición. Y por último hacemos la petición con axios, que en este caso usa el método '.post', al que le envía como parámetros la url y el objeto nuevo a insertar ('cliente'). Después de la promesa de axios cambiamos 'status' a true para usarlo como condición en la vista.

```
16     nuevoCliente = (e) => {
17         e.preventDefault();
18         var ced = this.cajaCedRef.current.value;
19         var nom = this.cajaNomRef.current.value;
20         var dir = this.cajaDirRef.current.value;
21         var ema = this.cajaEmaRef.current.value;
22         var tel = this.cajaTelRef.current.value;
23         var cliente = {
24             cedula: ced,
25             nombre: nom,
26             direccion: dir,
27             email: ema,
28             telefono: tel
29         };
30         var url = Global.urlclientes + '/clientes';
31         axios.post(url, cliente).then(res => {
32             this.setState({ status: true });
33         });
34     }
```

Por último, en el render mostraremos un formulario para recoger los datos del nuevo cliente. Cuando se envíe, llamará al método nuevoCliente para que inserte los datos. Tras el éxito de la petición se redirecciona al componente Clientes para visualizar la tabla con los cambios (el nuevo cliente estará añadido).

```
render() {
  if(this.state.status === true){
    return <Redirect to="/" />
  }
  return (
    <div>
      <h1>Nuevo Cliente</h1>
      <form onSubmit={this.nuevoCliente} style={{width: "50%", margin: "auto"}}>
        <label>Cédula: </label>
        <input type="text" name="cajaced" className="form-control" ref={this.cajaCedRef} />
        <label>Nombre: </label>
        <input type="text" name="cajanom" className="form-control" ref={this.cajaNomRef} />
        <label>Dirección: </label>
        <input type="text" name="cajadir" className="form-control" ref={this.cajaDirRef} />
        <label>Correo: </label>
        <input type="text" name="cajaema" className="form-control" ref={this.cajaEmaRef} />
        <label>Teléfono: </label>
        <input type="text" name="cajatel" className="form-control" ref={this.cajaTelRef} />
        <br />
        <button className="btn btn-success">Añadir</button>
      </form>
    </div>
  )
}
```

Al ejecutar el proyecto y seleccionar en el menú de navegación la opción de nuevo cliente aparece la siguiente pantalla:

Navbar
Cientes
Nuevo Cliente

Nuevo Cliente

Cédula:

Nombre:

Dirección:

Correo:

Teléfono:

Añadir

Al presionar el botón de Añadir se puede visualizar en la lista el cliente creado.

Navbar Clientes Nuevo Cliente								
Listado de Clientes								
Id Cliente	Cédula	Nombre	Dirección	Correo	Teléfono	Ver	Editar	Borrar
61a000572aab9029f47055b4	109812345	Sandra Milena Niño	Calle 9 # 12 - 37	sniño@lafloresta.edu.co	301245123	Detalles	Modificar	Eliminar
61a17d9381f3eb08e5531687	1098123451	Fernando de Jesús Alvarez	Calle 12 # 45 - 00	fjalvarez@floresta.edu.co	31256200	Detalles	Modificar	Eliminar
61a183b881f3eb08e5531688	63124561	María De Los Angeles Doña Florinda	Carrera 23 # 12 - 01	mangeles@floresta.edu.co	321456710	Detalles	Modificar	Eliminar
61a19e3381f3eb08e5531689	91213451	Juan José López	Carrera 23 # 45 - 33	plopez@floresta.edu.co	32145671	Detalles	Modificar	Eliminar
61a1ba5681f3eb08e553168a	91234561	Hernán Tarazona	Carrera 12 # 32 - 45	htarazona@lafloresta.edu.co	31267891	Detalles	Modificar	Eliminar
61a1baf081f3eb08e553168b	91567123	Pedro Antonio Zape	Calle 88 # 23 - 45	pzape@lafloresta.edu.co	321456712	Detalles	Modificar	Eliminar

6. Componente DetallesCliente.

Este componente sería muy similar al de 'Clientes, ya que requiere un método GET. Solo que esta vez añadiremos un parámetro a la ruta que será, en este caso, el id del cliente que la cual queremos obtener los detalles.

El proceso sería el mismo, y en el método mostrarCliente axios usará '.get' otra vez. Pero ahora, en la ruta que introducimos por parámetro añadimos un id: '/clientes/_id', '/clientes/_is'... Ese _id lo recibo por props.

```
src > components > CrudClientes > JS DetallesCliente.js > DetallesCliente > mostrarCliente
1  import React, { Component } from 'react';
2  import axios from 'axios';
3  import Global from '../Global';
4  import { NavLink } from 'react-router-dom';
5
6  export default class DetallesCliente extends Component {
7
8      state = {
9          cliente: {}
10         , status: false
11     }
12
13     mostrarCliente = () => {
14         var request = "/clientes/" + this.props._id;
15         var url = Global.urlclientes + request;
16         axios.get(url).then(res => {
17             this.setState({
18                 cliente: res.data
19                 ,status: true
20             });
21         });
22     }
23
24     componentDidMount = () => {
25         this.mostrarCliente();
26     }
27 }
```

```
render() {
  return (
    <div>
      <br /><br />
      <h1><u>Detalles del Cliente {this.props._id}</u></h1>
      {this.state.status === true &&
        (
          <React.Fragment>
            <br />
            <NavLink to="/" className="btn btn-sm btn-dark">Listado</NavLink>
            <br /><br />
            <h3>Nombre: <span (property) cliente: {}, fontWeight: "bold"></span>
              {this.state.cliente.nombre}</span></h3>
            <h3>Cédula: <span style={{color: "green", fontWeight: "bold"}}>
              {this.state.cliente.cedula}</span></h3>
            <h3>Correo: <span style={{color: "green", fontWeight: "bold"}}>
              [{this.state.cliente.email}]</span></h3>
            <NavLink to={"/update/" + this.state.cliente._id} className="btn btn-primary">
              Modificar</NavLink> &nbsp;&nbsp;&nbsp;
            <NavLink to={"/delete/" + this.state.cliente._id} className="btn btn-danger">
              Borrar</NavLink>
          </React.Fragment>
        )
      }
    </div>
  )
}
```

Vemos los detalles de nuestro cliente seleccionado.

Detalles del Cliente 61a183b881f3eb08e5531688

Listado

Nombre: **María De Los Angeles Doña Florinda**

Cédula: **63124561**

Correo: **mangeles@floresta.edu.co**

Modificar

Borrar

7. Componente UpdateCliente

Este componente sería muy similar al de InsertarCliente, ya que modificar un cliente requiere el mismo procedimiento que insertar, que es enviar un

objeto. Solo que en este caso el método que usa axios es '.put'.

Realizamos los imports necesarios y recogemos los datos nuevos a insertar. Y en el método modificarCliente hacemos la petición con axios, enviando por parámetros la url exacta (Global + petición) y el objeto 'cliente' nuevo.

```
src > components > CrudClientes > JS UpdateCliente.js > UpdateCliente > render
1  import React, { Component } from 'react';
2  import axios from 'axios';
3  import Global from '../Global';
4  import { Redirect, NavLink } from 'react-router-dom';
5
6  export default class UpdateCliente extends Component {
7
8      cajaCedRef = React.createRef();
9      cajaNomRef = React.createRef();
10     cajaDirRef = React.createRef();
11     cajaEmaRef = React.createRef();
12     cajaTelRef = React.createRef();
13
14     state = {
15         cliente: {},
16         status: false }
17 }
```

```
18     modificarCliente = (e) => {
19         e.preventDefault();
20         var ced = this.cajaCedRef.current.value;
21         var nom = this.cajaNomRef.current.value;
22         var dir = this.cajaDirRef.current.value;
23         var ema = this.cajaEmaRef.current.value;
24         var tel = this.cajaTelRef.current.value;
25         var cliente = {
26             cedula: ced,
27             nombre: nom,
28             direccion: dir,
29             email: ema,
30             telefono: tel
31         };
32         var request = "/clientes/" + this.props._id;
33         var url = Global.urlclientes + request;
34         axios.put(url, cliente).then(res => {
35             this.setState({status: true});
36         });
37     }
```

```

39     mostrarCliente = () => {
40         var request = "/clientes/" + this.props._id;
41         var url = Global.urlclientes + request;
42         axios.get(url).then(res => {
43             this.setState({
44                 cliente: res.data
45                 ,status: false
46             });
47         });
48     }
49
50     componentDidMount = () => {
51         this.mostrarCliente();
52     }
53

```

```

render() {
    if(this.state.status === true){
        return <Redirect to="/" />
    }
    return (
        <div>
            <h1>Modificar Cliente {this.props._id}</h1>
            <h2>{this.state.cliente.nombre}</h2>
            <NavLink to={'/detalles/' + this.props._id} className="btn btn-sm btn-dark">
                Detalles</NavLink>&nbsp;
            <NavLink to={'/'} className="btn btn-sm btn-dark">Lista</NavLink>
            <form onSubmit={this.modificarCliente} style={{width: "50%", margin: "auto"}}>
                <label>Cédula: </label>
                <input type="number" name="cajaced" className="form-control" ref={this.cajaCedRef}
                    defaultValue={this.state.cliente.cedula}/>
                <label>Nombre: </label>
                <input type="text" name="cajanom" className="form-control" ref={this.cajaNomRef}
                    defaultValue={this.state.cliente.nombre}/>
                <label>Dirección: </label>
                <input type="text" name="cajadir" className="form-control" ref={this.cajaDirRef}
                    defaultValue={this.state.cliente.direccion}/>
                <label>Correo: </label>
                <input type="text" name="cajaema" className="form-control" ref={this.cajaEmaRef}
                    defaultValue={this.state.cliente.email}/><br />
                <label>Teléfono: </label>
                <input type="text" name="cajatel" className="form-control" ref={this.cajaTelRef}
                    defaultValue={this.state.cliente.telefono} /><br />
                <button className="btn btn-success">Modificar</button>
            </form>
        </div>
    )
}

```

El resultado de seleccionar la opción de modificar se aprecia en la figura:

Modificar Cliente 61a19e3381f3eb08e5531689

Juan José López

[Detalles](#) [Lista](#)

Cédula:

91213451

Nombre:

Juan José López

Dirección:

Carrera 23 # 45 - 33

Correo:

plopez@floresta.edu.co

Teléfono:

32145671

[Modificar](#)

Al hacer una modificación y ejecutar con la opción de Modificar se actualiza el cliente.

8. Componente DeleteCliente.

Por último, el componente para eliminar cliente. También será similar a 'DetallesCliente', incluso con la misma url de la petición. Esta vez el método de axios será '.delete'.

```
src > components > CrudClientes > JS DeleteCliente.js > DeleteCliente > eliminarCliente
1  import React, { Component } from 'react';
2  import axios from 'axios';
3  import Global from '../Global';
4  import { Redirect, NavLink } from 'react-router-dom';
5
6  export default class DeleteCliente extends Component {
7
8      state = { status: false };
9
10     eliminarCliente = () => {
11         var request = "/clientes/" + this.props._id;
12         var url = Global.urlclientes + request;
13         axios.delete(url).then(res => {
14             this.setState({ status: true });
15         });
16     }
17 }
```

```
render() {  
    if(this.state.status === true){  
        return <Redirect to="/" />  
    }  
    return (  
        <div>  
            <br />  
            <h1 style={{color: "red"}}><?Desea eliminar la película {this.props._id}></h1><br />  
            <NavLink to="/" className="btn btn-light">Cancelar</NavLink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~</NavLink>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~</button>  
            </div>  
        )  
    }  
}
```

Navbar

Cientes

Nuevo Cliente

¿Desea eliminar el cliente 61a19e3381f3eb08e5531689?

Cancelar

Eliminar

Navbar

Cientes

Nuevo Cliente

Listado de Clientes

Id Cliente	Cédula	Nombre	Dirección	Correo	Teléfono	Ver	Editar	Borrar
61a000572aab9029f47055b4	109812345	Sandra Milena Niño	Calle 9 # 12 - 37	sniño@lafloresta.edu.co	301245123	Detalles	Modificar	Eliminar
61a17d9381f3eb08e5531687	1098123451	Fernando de Jesús Alvarez	Calle 12 # 45 - 00	fjalvarez@floresta.edu.co	31256200	Detalles	Modificar	Eliminar
61a183b881f3eb08e5531688	63124561	María De Los Angeles Doña Florinda	Carrera 23 # 12 - 01	mangeles@floresta.edu.co	321456710	Detalles	Modificar	Eliminar
61a1ba5681f3eb08e553168a	91234561	Hernán Tarazona	Carrera 12 # 32 - 45	htarazona@lafloresta.edu.co	31267891	Detalles	Modificar	Eliminar
61a1baf081f3eb08e553168b	91567123	Pedro Antonio Zape	Calle 88 # 23 - 45	pzape@lafloresta.edu.co	321456712	Detalles	Modificar	Eliminar