Servicio Nacional de Aprendizaje

Paso a paso de la creación de un React para consumir una API

David Felipe Ramírez Martin
Miguel Ángel Sarmiento Mojica
Kevin Alejandro Marín Hoyos

17/09/2023

TABLA DE CONTENIDOS

Introducción

El desarrollo web moderno ha evolucionado de manera significativa en los últimos años, y en el centro de esta revolución se encuentra React, una biblioteca de JavaScript desarrollada por Facebook. React se ha convertido en una herramienta esencial para aquellos que buscan crear aplicaciones web interactivas, dinámicas y altamente eficientes.

Este manual está diseñado para ser su guía en el mundo del desarrollo en React. Ya sea que sea un principiante que está dando sus primeros pasos en el desarrollo web o un desarrollador experimentado que busca mejorar sus habilidades, encontrará información valiosa y ejemplos prácticos que lo ayudarán a comprender y dominar React.

En las páginas que siguen, exploraremos en detalle cómo crear componentes, gestionar el estado de una aplicación, enlazar datos dinámicamente, y navegar entre diferentes vistas, entre otros aspectos esenciales. Aprenderemos a utilizar herramientas de desarrollo, como el ecosistema de React, para facilitar la gestión de la complejidad de las aplicaciones web modernas.

Así que, sin más preámbulos, ¡comencemos nuestro viaje hacia el fascinante mundo de React! Este manual le proporcionará los cimientos sólidos y la comprensión profunda que necesita para desarrollar aplicaciones web modernas y emocionantes con React.

Creación del proyecto

Instala los aplicativos necesarios.

Para el desarrollo de nuestra aplicación con React haremos uso de 3 aplicaciones esenciales, Visual Studio Code, NodeJS y un navegador tal como Google Chrome, Microsoft Edge, Opera GX, Etc.

Instalación Visual Studio Code

1. Descarga el instalador de Visual Studio Code desde la página oficial https://code.visualstudio.com/Download
2. Selecciona el sistema operativo que tienes instalado
3. Abre el .exe que se descargó en tu dispositivo (Imagen 1)
4. Acepta los términos y condiciones
5. Da click en siguiente
6. Da click en instalar

Instalación de navegador

1. Descarga el instalador del navegador https://www.google.com/intl/es_es/chrome/next-steps.html?brand=UUXU&statcb=1&installdataindex=empty&defaultbrowser=0
2. Abre el .exe que se descarga automáticamente (Imagen 2)
3. Da click en permitir
4. Termina la instalación
5. Abre la pestaña de aplicaciones de Google

6. Selecciona Chrome Web Store
7. Busca React Developer Tools
8. Abre la primera opción (Imagen 3)
9. Da click en añadir a Chrome

Instalación NodeJS

1. Descarga el instalador de NodeJS de la página oficial

   https://nodejs.org/es/download
2. Abre el .exe que se descarga automáticamente (Imagen 4)
3. Da click en permitir
4. Termina la instalación

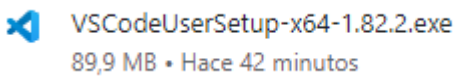

Imagen 1: Instalador de Visual Studio

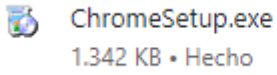

Imagen 2: Instalador de Chrome

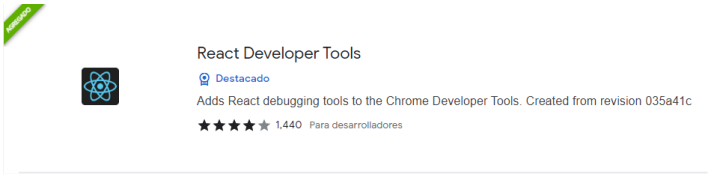

Imagen 3: Extensión de React para navegador

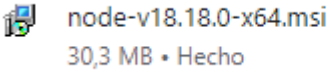

Imagen 4: Instalador NodeJS

Creación del proyecto

Crea una carpeta en la ubicación de tu preferencia

Abre el Visual Studio Code

Arrastra tu carpeta al área de trabajo de Visual Studio Code (Imagen 4)

Presiona las teclas win + ñ o en el menú superior crea una terminal nueva

Escribe los siguientes comandos en minúsculas y sin espacios

npx créate-react-app nombreaplicacion

npm i react-router-dom axios Bootstrap sweetalert2 sweetalert2-react-content

@fontawesome/fontawesome-free

Espera que se ejecuten correctamente ambos comandos (Imagen 5)

Prueba que se hayan ejecutado correctamente ejecutando los comandos

cd nombreaplicacion

npm start

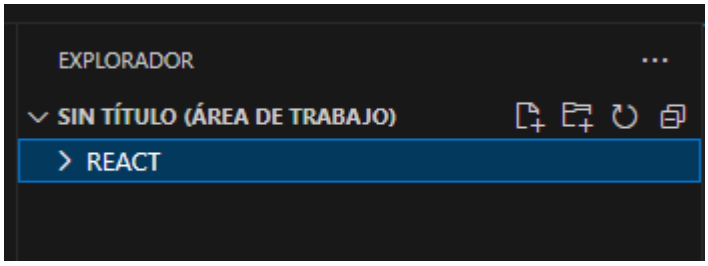

Imagen 4: Área de trabajo Visual Studio Code

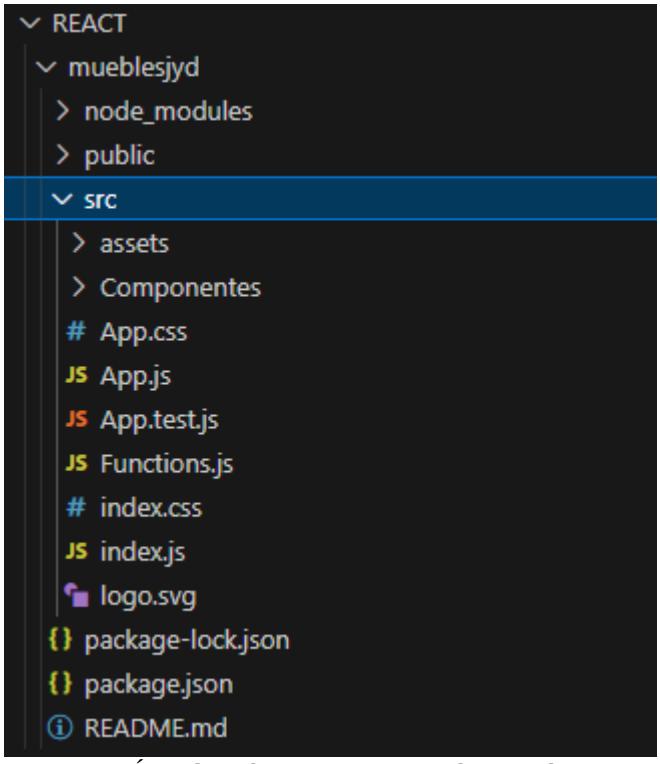

Imagen 5: Área de trabajo con React implementado

Creación de componentes

En el archivo app.js pon las respectivas dependencias

```
import {BrowserRouter, Routes, Route} from 'react-router-dom'
import ShowJobs from './Componentes/ShowJobs'
```

Dentro del contenido de la función App pon las siguientes líneas

```
<BrowserRouter>
      <Routes>
       <Route path='/' exact element={<ShowJobs/>}></Route>
</Routes>
</BrowserRouter>
```

Da click derecho sobre la carpeta src y crea una carpeta llamada componentes

Crea un archivo en la carpeta componentes llamado ShowJobs.js

Importa las librerías respectivas

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import Swal from "sweetalert2";
import withReactContent from "sweetalert2-react-content";
```

```
import { show_alert } from "../Functions
```

Crea una constante con una función Lambda de la siguiente manera const

```
const ShowJobs = () => {
```

Crea constantes con los nombres de la tabla de la base de datos que usara tu API

Ej:

```
const url = "endpoint de tu API";
const [campo1, setCampo1] = useState();
const [campo2, setCampo2] = useState();
const [cargos, setCargos] = useState([]);
const [operation, setOperation] = useState([1]);
const [title, setTitle] = useState();
const [idToEdit, setidToEdit] = useState(null);
```

Crea una función llamada useEffect de la siguiente manera:

```
useEffect(() => {
  getJobs();
}, []);
```

Define una función asíncrona llamada getJobs con el siguiente contenido:

```
const getJobs = async () => {
  const respuesta = await axios.get(url);
  setCargos(respuesta.data);
};
```

Define una función lambda llamada openModal con el siguiente contenido:

```
const openModal = (op, nombreCargo, estado, id) => {
  setEstado("");
  setOperation(op);
  if (op === 1) {
    setTitle("Registrar cargo");
    setNombreCargoActual("");
    setEstado("");
  } else if (op === 2) {
    setTitle("Editar cargo");
    setNombreCargoActual(nombreCargo);
    setEstado(estado);
    setidToEdit(id);
  }
  window.setTimeout(function () {
    document.getElementById("nombreCargo").focus();
 }, 500);
};
```

Define una función lambda con una constante con el siguiente contenido:

```
  const validar = (id) => {
    var parametros;
    var metodo;

    if (NombreCargo.trim() === "") {
      show_alert("error", "Escribe el nombre del cargo");
    } else if (estado === "") {
      show_alert("error", "Escribe el estado del cargo");
    } else {
      if (operation === 1) {
        parametros = { NombreCargo: NombreCargo, estado:
estado };
        metodo = "POST";
      } else {
        parametros = { NombreCargo: NombreCargo, estado:
estado };
        console.log(parametros);
        metodo = "PUT";
      }

      enviarSolicitud(metodo, parametros, id);
    }
  };
```

Define una constante con una función asíncrona con el siguiente contenido:

```
const enviarSolicitud = async (metodo, parametros, id) => {
  if (metodo === "POST") {
    axios
      .post(`${url}`, parametros)
      .then(function (respuesta) {
        show_alert("success", "Cargo creado");
        document.getElementById("btnCerrar").click();
        getJobs();
      })
      .catch(function (error) {
        show_alert("error", "Error de solucitud");
        console.log(error);
      });
  } else if (metodo === "PUT") {
    axios
      .put(`${url}${id}`, parametros)
```

```
      .then(function (respuesta) {
        console.log("Solicitud PUT exitosa:",
respuesta.data);
        var tipo = respuesta.data[0];
        var msj = respuesta.data[1];
        show_alert("success", "Cargo editado con exito");
        document.getElementById("btnCerrar").click();
        getJobs();
      })
      .catch(function (error) {
        show_alert("Error de solucitud", "error");
        console.log(error);
      });
  }
};
```

Define una constante con una función asíncrona con el siguiente contenido:

Dd

```
const deleteCargo = (id, name) => {
  const MySwal = withReactContent(Swal);
  MySwal.fire({
    title: "¿Seguro quieres eliminar el cargo " + name +
"?",
    icon: "question",
    text: "No se podra dar marcha atras",
    showCancelButton: true,
    confirmButtonText: "Si, eliminar",
    cancelButtonText: "Cancelar",
  }).then(async (result) => {
    if (result.isConfirmed) {
      try {
        console.log(`${url}${id}`);
        await axios.delete(`${url}${id}`);
        show_alert("success", "Cargo eliminado
exitosamente");
        getJobs();
      } catch (error) {
        show_alert("error", "Error al eliminar el cargo");
        console.error(error);
      }
    } else {
      show_alert("info", "El cargo no fue eliminado");
```

```
    }
  });
};
```

Creación del Functions

Se Crea un archivo de tipo JS llamado Functions

En el archivo Functions.js se importan las librerías que se deseen usar, en este caso

```
import Swal from "sweetalert2";
import withReactContent from "sweetalert2-react-content";
```

Dentro del contenido del archivo Functions se agrega una línea que permitirá personalizar la alerta de sweetalert

```
export function show_alert(icono, mensaje, foco = ""){}
```

Dentro de la nueva línea entre las llaves se agregaran las instrucciones para su funcionamiento

```
{
  OnFocus(foco);

  const MySwal = withReactContent(Swal);

  MySwal.fire({
    title: mensaje,
    icon: icono,
  });
}
```

Se añade una función OnFocus que permitirá saber a qué campo se está referenciando al momento de mandar una alerta o estar realizando una operación que necesite saber esta información

```
function OnFocus(foco) {
  if (foco !== "") {
    document.getElementById("foco").focus();
  }
```

```
}
```

Creacion Return

Se crea un return para darle el estilo a la pestaña de ShowJobs

```
return ()
```

Dentro del return se añadirán las indicaciones que permitirán crear la barra de navegación superior

```
<div className="App">
      <div id="custom-container" className="d-flex">
      <nav className="navbar navbar-light navbar-expand-md
py-3" style={{ width: '100%', background: '#002b5b' }}>
          <div className="container">
            <img src={Logo} alt="Logo" />
            <button className="navbar-toggler" data-bs-
toggle="collapse" data-bs-target="#navcol-2">
              <span className="visually-hidden">Toggle
navigation</span>
              <span className="navbar-toggler-icon"></span>
            </button>
            <div className="collapse navbar-collapse"
id="navcol-2" style={{ color: 'var(--bs-black)', fontSize:
'20px' }}>
              <ul className="navbar-nav ms-auto">
                <li className="nav-item"><a className="nav-
link active" data-bss-hover-animate="flash" href="/"
style={{ color: 'var(--bs-white)', fontWeight: 'bold',
fontFamily: 'ABeeZee, sans-serif' }}>Inicio</a></li>
                <li className="nav-item" data-bss-hover-
animate="flash" style={{ color: 'var(--bs-white)',
fontWeight: 'bold' }}><a className="nav-link" data-bss-
hover-animate="flash" href="./Productos" style={{ color:
'var(--bs-white)', fontWeight: 'bold', fontFamily: 'ABeeZee,
sans-serif' }}>Productos</a></li>
                <li className="nav-item" data-bss-hover-
animate="flash" style={{ color: 'var(--bs-white)',
fontWeight: 'bold' }}><a className="nav-link" data-bss-
hover-animate="flash" href="./QuienesSomos" style={{ color:
```

```
'var(--bs-white)', fontWeight: 'bold', fontFamily: 'ABeeZee,
sans-serif' }}>¿Quiénes somos?</a></li>
              </ul>
              <ul className="navbar-nav" data-bss-hover-
animate="flash" style={{ color: 'var(--bs-white)',
fontWeight: 'bold' }}>
                <li className="nav-item" data-bss-hover-
animate="flash" style={{ color: 'var(--bs-white)',
fontWeight: 'bold' }}></li>
                <li className="nav-item" data-bss-hover-
animate="flash" style={{ color: 'var(--bs-white)',
fontWeight: 'bold' }}><a className="nav-link" data-bss-
hover-animate="flash" href="./Registrarse" style={{ color:
'var(--bs-white)', fontWeight: 'bold', fontFamily: 'ABeeZee,
sans-serif' }}>Registrarse</a></li>
                <li className="nav-item" data-bss-hover-
animate="flash" style={{ color: 'var(--bs-white)',
fontWeight: 'bold' }}><a className="nav-link" data-bss-
hover-animate="flash" href="./LogIn" style={{ color: 'var(--
bs-white)', fontWeight: 'bold', fontFamily: 'ABeeZee, sans-
serif' }}>Iniciar sesión</a></li>
              </ul>
            </div>
          </div>
        </nav>
      </div>
```

Posteriormente se crea el apartado que muestra una tabla con todos los cargos registrados

```
<div className="container-fluid">
        <div className="row mt-3">
          <div className="col-md-4 offset-md-4">
            <div className="d-grid mx-auto"></div>
          </div>
        </div>

        <div className="container-fluid custom-container">
          <div className="card" id="TableSorterCard-1">
```

```
            <div className="card-header py-3 custom-card-
header">
              <div className="row table-topper align-items-
center">
                <div className="col-12 col-sm-5 col-md-6
text-start custom-title">
                  <h1 className="custom-title">Cargos</h1>
                </div>
                <div className="col-12 col-sm-7 col-md-6
text-end custom-button">
                  <button
                    onClick={() => openModal(1)}
                    className="btn btn-primary"
                    data-bs-toggle="modal"
                    data-bs-target="#modalCargos"
                  >
                    <i className="fa-solid fa-circle-
plus"></i> Añadir cargo
                  </button>
                </div>
              </div>
            </div>
            <div className="row">
              <div className="col-12">
                <div className="table-responsive custom-
table-responsive">
                  <table
                    className="table table-striped table
tablesorter custom-table"
                    id="ipi-table"
                  >
                    <thead>
                      <tr>
                        <th>#</th>

                        <th>Nombre</th>
                        <th>Estado</th>
                        <th>Botones</th>
                      </tr>
```

```
                    </thead>
                    <tbody className="text-center">
                      {cargos.map((cargo, i) => (
                        <tr key={cargo.id}>
                          <td>{i + 1}</td>

                          <td>{cargo.nombreCargo}</td>
                          <td>{cargo.estado}</td>
                          <td>
                            <button
                              className="btn btnMaterial
btn-flat success semicircle"
                              data-bs-toggle="modal"
                              data-bs-target="#modalCargos"
                              onClick={() =>
                                openModal(
                                  2,
                                  cargo.nombreCargo,
                                  cargo.estado,
                                  cargo.id
                                )
                              }
                            >
                              <i className="fa-solid fa-edit
btn-editar"></i>
                            </button>
                            <button
                              onClick={() =>
                                deleteCargo(cargo.id,
cargo.nombreCargo)
                              }
                              className="btn btnMaterial
btn-flat accent btnNoBorders checkboxHover"
                            >
                              <i className="fa-solid fa-
trash btn-eliminar"></i>
                            </button>
                          </td>
                        </tr>
```

```
                    ))}
                  </tbody>
                </table>
              </div>
            </div>
          </div>
        </div>
      </div>
```

Seguidamente dentro del return se crea el pie de página que contendrá los términos y condiciones, las políticas de privacidad y enlaces a las redes sociales

```
<div className="text-center py-4 custom-background">
        <div className="container">
          <div className="row row-cols-1 row-cols-lg-3">
            <div className="col custom-color">
              <p className="my-2 custom-p">Copyright © 2023
Brand</p>
            </div>
            <div className="col custom-color">
              <ul className="list-inline my-2 custom-list">
                <li className="list-inline-item me-4 custom-
list-item">
                  <div className="bs-icon-circle bs-icon-
primary bs-icon">
                    <svg
                      xmlns="http://www.w3.org/2000/svg"
                      width="1em"
                      height="1em"
                      fill="currentColor"
                      viewBox="0 0 16 16"
                      className="bi bi-facebook custom-icon"
                    >
                      <path d="M16 8.049c0-4.446-3.582-8.05-
8-8.05C3.58 0-.002 3.603-.002 8.05c0 4.017 2.926 7.347 6.75
7.951v-5.625h-2.03V8.05H6.75V6.275c0-2.017 1.195-3.131
3.022-3.131.876 0 1.791.157 1.791.157v1.98h-1.009c-.993 0-
1.303.621-1.303 1.258v1.51h2.218l-.354 2.326H9.25V16c3.824-
.604 6.75-3.934 6.75-7.951z"></path>
```

```
                    </svg>
                  </div>
                </li>
                <li className="list-inline-item me-4 custom-
list-item">
                  <div className="bs-icon-circle bs-icon-
primary bs-icon">
                    <svg
                      xmlns="http://www.w3.org/2000/svg"
                      width="1em"
                      height="1em"
                      fill="currentColor"
                      viewBox="0 0 16 16"
                      className="bi bi-twitter custom-icon"
                    >
                      <path d="M5.026 15c6.038 0 9.341-5.003
9.341-9.334 0-.14 0-.282-.006-.422A6.685 6.685 0 0 0 16
3.542a6.658 6.658 0 0 1-1.889.518 3.301 3.301 0 0 0 1.447-
1.817 6.533 6.533 0 0 1-2.087.793A3.286 3.286 0 0 0 7.875
6.03a9.325 9.325 0 0 1-6.767-3.429 3.289 3.289 0 0 0 1.018
4.382A3.323 3.323 0 0 1 .64 6.575v.045a3.288 3.288 0 0 0
2.632 3.218 3.203 3.203 0 0 1-.865.115 3.23 3.23 0 0 1-.614-
.057 3.283 3.283 0 0 0 3.067 2.277A6.588 6.588 0 0 1 .78
13.58a6.32 6.32 0 0 1-.78-.045A9.344 9.344 0 0 0 5.026
15z"></path>
                    </svg>
                  </div>
                </li>
                <li className="list-inline-item custom-list-
item">
                  <div className="bs-icon-circle bs-icon-
primary bs-icon">
                    <svg
                      xmlns="http://www.w3.org/2000/svg"
                      width="1em"
                      height="1em"
                      fill="currentColor"
                      viewBox="0 0 16 16"
                      className="bi bi-instagram custom-
icon"
```

```
                    >
                      <path d="M8 0C5.829 0 5.556.01
4.703.048 3.85.088 3.269.222 2.76.42a3.917 3.917 0 0 0-
1.417.923A3.927 3.927 0 0 0 .42 2.76C.222 3.268.087 3.85.048
4.7.01 5.555 0 5.827 0 8.001c0 2.172.01 2.444.048
3.297.04.852.174 1.433.372 1.942.205.526.478.972.923
1.417.444.445.89.719 1.416.923.51.198 1.09.333
1.942.372C5.555 15.99 5.827 16 8 16s-.01-2.445-.048-3.299c-
.04-.851-.175-1.433-.372-1.941a2.47 2.47 0 0 1-.599-.919c-
.28-.28-.546-.453-.92-.598-.28-.11-.704.24-1.485.276-
.843.038-1.096.047-3.232.047s-2.39-.009-3.233-.047c-.78-
.036-1.203-.166-1.485-.276a2.478 2.478 0 0 1-.92-.598 2.48
2.48 0 0 1-.6-.92c-.109-.281-.24-.705-.275-1.485-.038-.843-
.046-1.096-.046-3.233 0-2.136.008-2.388.046-3.231.036-
.78.166-1.204.276-1.486.145-.373.319-.64.599-.92.28-.28.546-
.453.92-.598.282-.11.705-.24 1.485-.276.738-.034 1.024-.044
2.515-.045v.002zm4.988 1.328a.96.96 0 1 0 0 1.92.96.96 0 0 0
0-1.92zm-4.27 1.122a4.109 4.109 0 1 0 0 8.217 4.109 4.109 0
0 0 0-8.217zm0 1.441a2.667 2.667 0 1 1 0 5.334 2.667 2.667 0
0 1 0-5.334z"></path>
                    </svg>
                  </div>
                </li>
              </ul>
            </div>
            <div className="col custom-color">
              <ul className="list-inline my-2 custom-list">
                <li className="list-inline-item">
                  <p className="custom-link">Privacy
Policy</p>
                </li>
                <li className="list-inline-item">
                  <p className="custom-link">Terms of
Use</p>
                </li>
              </ul>
            </div>
          </div>
        </div>
      </div>
```

Por último se crea un apartado en el return que permitirá manejar la ventana emergente que permite crear nuevos cargos

```
<div id="modalCargos" className="modal fade" aria-
hidden="true">
        <div className="modal-dialog">
          <div className="modal-content">
            <div className="modal-header">
              <label className="h5">{title}</label>
              <button
                type="button"
                className="btn-close"
                data-bs-dismiss="modal"
                aria-label="close"
              ></button>
            </div>
            <div className="modal-body">
              <input type="hidden" id="id"></input>
              <div className="input-group mb-3">
                <span className="input-group-text">
                  {" "}
                  <i className="fa fa-user"></i>
                </span>
                <input
                  type="text"
                  id="nombreCargo"
                  className="form-control"
                  placeholder="Nombre"
                  value={NombreCargo}
                  onChange={(e) =>
setNombreCargoActual(e.target.value)}
                ></input>
              </div>
              <div className="input-group mb-3">
                <span className="input-group-text">
                  {" "}
                  <i className="fa fa-power-off"></i>
                </span>
                <input
```

```
                    type="text"
                    id="estado"
                    className="form-control"
                    placeholder="Estado"
                    value={estado}
                    onChange={(e) =>
setEstado(e.target.value)}
                  ></input>
                </div>
                <div className="d-grid col-6 mx-auto">
                  <button
                    onClick={() => validar(idToEdit,
NombreCargo)}
                    className="btn btn-success"
                  >
                    <i className="fa-solid fa-floppy-
disk"></i> Guardar
                  </button>
                </div>
              </div>
              <div className="modal-footer">
                <button
                  id="btnCerrar"
                  type="button"
                  className="btn btn-secondary"
                  data-bs-dismiss="modal"
                >
                  Cerrar
                </button>
              </div>
            </div>
          </div>
        </div>
```

Explicacion funciones

Importar librerías

import React, { useState, useEffect } from "react";

```
import axios from "axios";
import Swal from "sweetalert2";
import withReactContent from "sweetalert2-react-content";
```

En esta sección, se importan las librerías necesarias para el componente. React es la biblioteca principal para crear componentes, useState y useEffect son hooks de React para gestionar el estado y el ciclo de vida del componente, axios se utiliza para hacer solicitudes HTTP, y Swal es una librería para mostrar ventanas modales.

Declaración del componente ShowJobs:

```
const ShowJobs = () => {
  // Estado y variables
  const url = "https://localhost:7284/api/cargos/";
  const [id, setId] = useState();
  const [cargos, setCargos] = useState([]);
  const [NombreCargo, setNombreCargoActual] = useState("");
  const [estado, setEstado] = useState();
  const [operation, setOperation] = useState([1]);
  const [title, setTitle] = useState();
  const URL = `https://localhost:7284/api/cargos/${id}`;
  const [idToEdit, setidToEdit] = useState(null);

  // Efecto para cargar los cargos al montar el componente
```

```
useEffect(() => {
  getJobs();
}, []);
```

Aquí se declaran varias variables de estado utilizando el hook useState. Estas variables de estado se utilizan para almacenar datos como id, cargos, NombreCargo, estado, operation, title, y idToEdit. También se declara la variable url que almacena la URL de la API que se va a utilizar. Además, se utiliza el hook useEffect para llamar a la función getJobs cuando el componente se monta por primera vez (con un array vacío de dependencias, lo que significa que solo se ejecuta una vez).

Función getJobs:

```
const getJobs = async () => {
  const respuesta = await axios.get(url);
  setCargos(respuesta.data);
};
```

Esta función utiliza axios para hacer una solicitud GET a la URL especificada y luego actualiza el estado cargos con los datos de respuesta.

Función openModal:

```
const openModal = (op, nombreCargo, estado, id) => {
```

```
        // Configuración de modal y actualización de estados
  // …
}
```

Esta función se utiliza para abrir un modal (ventana emergente) para registrar o editar un cargo. Dependiendo de la operación (op), se configuran los estados operation, title, NombreCargo, estado, y idToEdit para mostrar la información correcta en el modal.;

Función validar:

```
const validar = (id) => {
  // Validación de datos y envío de solicitud (POST o PUT)
  // ...
};
```

Esta función se utiliza para validar los datos ingresados en el formulario del modal y luego enviar una solicitud HTTP (ya sea POST o PUT) según la operación especificada.

Función enviarSolicitud:

```
const enviarSolicitud = async (metodo, parametros, id) => {
  // Manejo de solicitudes POST o PUT
  // ...
```

```
};
```

Esta función se utiliza para enviar una solicitud HTTP (ya sea POST o PUT) a la API utilizando axios. Dependiendo del método, se realiza la solicitud correspondiente y se manejan las respuestas.

Función deleteCargo:

```
const deleteCargo = (id, name) => {
  // Confirmación de eliminación y envío de solicitud DELETE
  // ...
};
```

Esta función se utiliza para mostrar un mensaje de confirmación antes de eliminar un cargo. Si el usuario confirma, se envía una solicitud HTTP DELETE para eliminar el cargo correspondiente.

En resumen, el código implementa un componente de React para gestionar cargos, que incluye funciones para mostrar un modal de registro/edición, validar datos, enviar solicitudes HTTP para crear o editar cargos, y eliminar cargos con confirmación del usuario. Además, se carga la lista de cargos al montar el componente utilizando una solicitud GET a una API.

Conclusión

En resumen, este manual proporciona una guía completa y detallada para la creación de un proyecto en React que consuma una API de manera efectiva. A lo largo de este manual, hemos explorado los conceptos fundamentales de React, la configuración del entorno de desarrollo, la gestión de componentes y el uso de axios para realizar solicitudes a la API. Además, hemos aprendido a manejar datos recibidos de la API y a actualizar dinámicamente nuestra interfaz de usuario. Con esta sólida base de conocimientos, estás listo para desarrollar proyectos web interactivos y dinámicos que aprovechen todo el potencial de React y las API externas. ¡Adelante y comienza a crear aplicaciones web modernas y poderosas!