Buscar un Pokémon

Estefanía Becerra Quintero

C.C: 1004717642

Docente: José Nicolás Aristizábal Ramírez

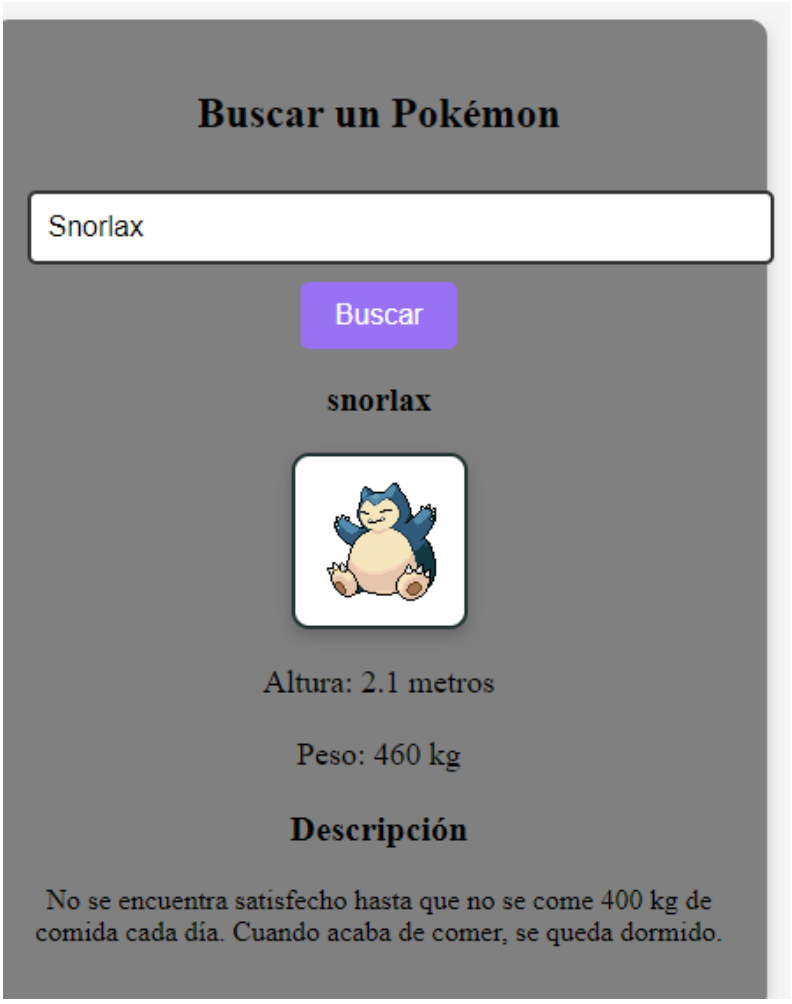
Corporación Instituto de Administración y Finanzas, CIAF.

Ingeniería de Software

2023

Introducción:

El código presentado representa una aplicación de búsqueda de Pokémon desarrollada con React, que utiliza una variedad de hooks de React para lograr funcionalidades esenciales. Esta aplicación permite a los usuarios buscar información detallada sobre Pokémon utilizando la API de PokeAPI. A continuación, se proporciona un desglose detallado de cómo funciona el código.



Fuente: Elaboración propia. Librería Axios de JavaScript

PokerAPI

Aquí está el código explicado línea por línea:

```
import React, {
 useState,
 useEffect,
 useLayoutEffect,
 useRef,
 useCallback
} from "react";
import axios from "axios";
import "./App.css";
```

Importamos las funciones y módulos necesarios de React y otras bibliotecas como axios. También importamos un archivo CSS para los estilos.

```
function BuscarPokemon() {
  const [terminoBusqueda, setTerminoBusqueda] = useState("");
  const [datosPokemon, setDatosPokemon] = useState(null);
  const [backgroundColor, setBackgroundColor] = useState("transparent");
  const [historia, setHistoria] = useState("");
```

Aquí definimos el componente BuscarPokemon como una función de React. Utilizamos varios estados con el hook useState para almacenar información en el componente. terminoBusqueda almacena el término de búsqueda del usuario, datosPokemon almacena la información del Pokémon, backgroundColor controla el color de fondo y historia almacena la descripción del Pokémon.

```
  function getRandomColor() {
    const letters = "0123456789ABCDEF";
    let color = "#";
```

```
  for (let i = 0; i < 6; i++) {
    color += letters[Math.floor(Math.random() * 16)];
  }
  return color;
}
```

Se define una función getRandomColor que genera un color hexadecimal aleatorio. Esta función se usará para cambiar el color de fondo del botón de búsqueda.

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

Se utiliza el hook useEffect para cargar datos iniciales cuando el componente se monta. En este caso, llama a la función cargarDatosIniciales para establecer información de ejemplo de un Pokémon llamado "Pikachu" y una descripción ficticia.

```
const cargarDatosIniciales = () => {
  const datosIniciales = {
    name: "Pikachu",
    sprites: { front_default: "url_de_la_imagen" },
    height: 40,
    weight: 60
  };

  // Descripción ficticia
  const descripcionInicial =
    "Pikachu es un Pokémon eléctrico muy conocido por su poderoso ataque de rayo.";
```

```
  setDatosPokemon(datosIniciales);
  setHistoria(descripcionInicial);
};
```

La función cargarDatosIniciales establece información de ejemplo en los estados datosPokemon y historia. Esta información incluye el nombre del Pokémon, una URL de imagen, altura y peso, así como una descripción ficticia.

```
const botonBusquedaRef = useRef(null);
```

Se utiliza el hook useRef para crear una referencia al botón de búsqueda. Esta referencia se utilizará para acceder al botón en el DOM y cambiar su estilo.

```
const obtenerDescripcionPokemon = (id) => {
  axios
    .get(`https://pokeapi.co/api/v2/pokemon-species/${id}/`)
    .then((response) => {
      const descripcion = response.data.flavor_text_entries.find(
        (entry) => entry.language.name === "es"
      );
      if (descripcion) {
        setHistoria(descripcion.flavor_text);
      }
    })
    .catch((error) => {
      console.error("Error al obtener la descripción:", error);
    });
};
```

La función obtenerDescripcionPokemon realiza una solicitud a la API de PokeAPI para obtener la descripción del Pokémon en español. Utiliza el ID del Pokémon para hacer la solicitud y, si se encuentra una descripción, la establece en el estado historia.

```
const manejarBusqueda = useCallback(() => {
  if (terminoBusqueda.trim() === "") {
    alert("Ingresa un nombre o ID de Pokémon");
    return;
  }

  axios
    .get(`https://pokeapi.co/api/v2/pokemon/${terminoBusqueda.toLowerCase()}`)
    .then((response) => {
      setDatosPokemon(response.data);
      botonBusquedaRef.current.style.backgroundColor = getRandomColor();
      setBackgroundColor("gray");
      // Obtener descripción del Pokémon
      obtenerDescripcionPokemon(response.data.id);
    })
    .catch((error) => {
      console.error("Error:", error);
      setDatosPokemon(null);
    });
}, [terminoBusqueda]);
```

La función manejarBusqueda se utiliza para manejar la búsqueda de un Pokémon. Verifica si se ha ingresado un término de búsqueda válido. Luego, realiza una solicitud a la API de PokeAPI para obtener información sobre el Pokémon, establece los datos en el estado datosPokemon, cambia el color de fondo del botón de búsqueda y llama a obtenerDescripcionPokemon para obtener la descripción del Pokémon.

```
useLayoutEffect(() => {
  if (datosPokemon) {
```

```
    const randomColor = getRandomColor();
    botonBusquedaRef.current.style.backgroundColor = randomColor;
  }
 }, [datosPokemon]);
```

Se utiliza el hook useLayoutEffect para cambiar el color de fondo del botón de búsqueda una vez que se han obtenido los datos del Pokémon. Esto se hace eligiendo un color aleatorio y aplicándolo al botón.

```
 return (
  <div className="container">
   <div className="section" style={{ backgroundColor }}>
    <h2>Buscar un Pokémon</h2>
    <input
     type="text"
     placeholder="Ingresa el nombre o ID de un Pokémon"
     value={terminoBusqueda}
     onChange={(e) => setTerminoBusqueda(e.target.value)}
    />
    <button ref={botonBusquedaRef} onClick={manejarBusqueda}>
     Buscar
    </button>
    {datosPokemon && (
     <div>
      <h3>{datosPokemon.name}</h3>
      <img
       src={datosPokemon.sprites.front_default}
       alt={datosPokemon.name}
       className="pokemon-image"
      />
      <p className="pokemon-info">
```

```
            Altura: {datosPokemon.height / 10} metros
          </p>
          <p className="pokemon-info">Peso: {datosPokemon.weight / 10} kg</p>
        </div>
      )}
      {historia && (
        <div>
          <h3 className="description-title">Descripción</h3>
          <p className="pokemon-description">{historia}</p>
        </div>
      )}
    </div>
  </div>
 );
}
```

El componente renderiza una interfaz de búsqueda de Pokémon. Muestra un campo de entrada de texto, un botón de búsqueda, el nombre del Pokémon, una imagen, información de altura

Elementos de Interfaz de Usuario:

La interfaz de usuario incluye un campo de entrada de texto para el término de búsqueda, un botón de búsqueda y una sección que muestra los datos del Pokémon encontrado, incluyendo su nombre, imagen, altura y peso. Además, se muestra la descripción del Pokémon en una sección separada.

Conclusión:

El código de esta aplicación de búsqueda de Pokémon demuestra cómo utilizar hooks de React, como useState, useEffect, useRef, useCallback y useLayoutEffect, para crear una aplicación interactiva y dinámica.