educación IT
CODE YOUR FUTURE

ReactJS Developer

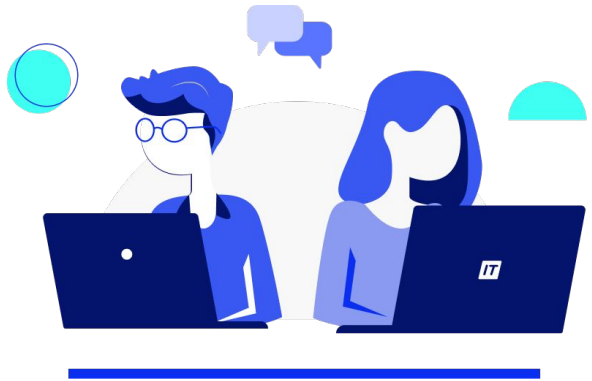
Módulo 4

Hooks

Hooks en React Router

Los *hooks* de React Router permiten ir añadiendo **características puntuales a un componente funcional.**

Como es de esperarse, estas características están relacionadas **con la comunicación con la ruta actual.** Vamos a ver los *hooks* más comunes en el desarrollo con React-Router.



UseRoutes

El *hook* **`useRoutes`** es un reemplazo al **`Routes`** y **`Route`**. Sirve para implementar la misma lógica que esos dos componentes pero de forma **programática** en JavaScript (en vez de la forma declarativa con *JSX*).

Este *hook* recibe como argumento un `array` de objetos. Cada objeto puede tener **las mismas propiedades que un `Route`,** incluyendo la propiedad **`Children`**. Piensa en este objeto como las *props* que se le pasarían a un `Route`.

El hook `useRoutes` retorna **un elemento de React listo para ser renderizado en *JSX* o retornado directamente por un componente.**

```jsx
import { useRoutes } from "react-router-dom";

function App() {
  let routes = useRoutes([
    {
      path: "/",
      element: <Dashboard />,
      children: [
        {
          path: "messages",
          element: <DashboardMessages />
        },
        { path: "tasks", element: <DashboardTasks /> }
      ]
    },
    { path: "team", element: <AboutPage /> }
  ]);

  return routes;
}

export default App;
```

UseNavigate

El *hook* **`useNavigate`** retorna **una función que hace exactamente lo mismo que el componente `Navigate`.**

Nos puede ser muy útil cuando necesitamos realizar **una redirección de forma programática en lugar de *JSX*** (como por ejemplo en el *handler* de un evento).

```jsx
import { useNavigate } from "react-router-dom";

function SignupForm() {
  let navigate = useNavigate();

  async function handleSubmit(event) {
    event.preventDefault();
    await submitForm(event.target);
    navigate("../success", { replace: true });
  }

  return <form onSubmit={handleSubmit}>{/* ... */}</form>;
}

export default SignupForm;

```

UseParams

El *hook* **`useParams`** permite **acceder a los parámetros pasados por la *URL*** ¿Recuerdas que, cuando vimos el `Route`, comentamos que se pueden pasar parámetros anteponiendo dos puntos (`:`) en un segmento de la *URL*? Este *hook* es la otra cara de la moneda. Nos permite obtener dichos parámetros **en un componente.**

El *hook* retorna un objeto JavaScript plano donde las claves son **los nombres definidos en la ruta** (los segmentos que empiecen con dos puntos) y **los valores son los caracteres ubicados en esa posición de la ruta.**

Por ejemplo

Si hay un `Route` que busca

`“/product/:nombre”`

y se accede a la ruta

`“/product/anteojos-redondeados”`,

el valor devuelto por `useParams()` será

`{ nombre: “anteojos-redondeados” }`.

```jsx
import { useEffect, useState } from 'react';
import { Routes, Route, useParams } from 'react-router-dom';

function ProfilePage() {
  let { userId } = useParams();
  const [user, setUser] = useState({});

  useEffect(() => {
    fetch(`/users/${userId}`)
        .then(r => r.json())
        .then(d => setUser(d))
  });

  return (
      <div>{user.name}</div>
  );
}

function App() {
  return (
    <Routes>
      <Route path="users">
        <Route path=":userId" element={<ProfilePage />} />
      </Route>
    </Routes>
  );
}

export default App;
```

UseSearchParams

Muchas veces necesitamos realizar búsquedas. Para eso es muy útil el *query string* de la *URL*. El ***query string*** son **todos los datos pasados como valores GET** (aquellos que se encuentran luego de un signo "**?**" en la *URL*).

El *query string* permite **realizar búsquedas sin cambiar de *URL*,** por lo que son muy usados. Con este *hook* podemos leer dichos parámetros y usarlos para mejorar nuestras búsquedas. Este *hook* retorna una instancia de URLSearchParams basada en el ***query string* actual.**

```jsx
import { useSearchParams } from "react-router-dom";

function ProductList(props) {
    let [searchParams, setSearchParams] = useSearchParams(),
        products = props.products || props.children || [];

    const productQuery = searchParams.get("q");

    products = products.filter((p) =>
        p.name.toLowerCase().includes(productQuery.toLowerCase())
    );

    return (
        <div>
            {products.map((p, i) => (
                <div>{p.name}</div>
            ))}
        </div>
    );
}

export default ProductList;
```

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!