Temario:

¿Qué es React Router?

Instalación

Configuración de React-Router

Tipos de rutas

UseParams

¿Qué es React Router?

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

Beneficios de React Router:

- Establecer rutas en nuestra aplicación ej: Home, About, User.
- Realizar redirecciones
- Acceso al historial del navegador
- Manejo de rutas con parámetros
- Páginas para el manejo de errores como 404
- Componentes pilares de React Router
- BrowserRouter

Este componente es el encargado de envolver nuestra aplicación dándonos acceso al API historial de HTML5 (pushState, replaceState y el evento popstate) para mantener su UI sincronizada con la URL.

Switch

Este componente es el encargado de que solo se renderice el primer hijo Route o Redirect que coincide con la ubicación. Si no usar este componente todos los componentes Route o Redirect se van a renderizar mientras cumplan con la condición establecida.

Route

Con Route podemos definir las rutas de nuestra aplicación, quizás sea el componente más importante de React Router para llegar a comprender todo el manejo de esta librería. Cuandodefinimos una ruta con Route le indicamos que componente debe renderizar.

Este componente cuanta con algunas propiedades

Path: la ruta donde debemos renderizar nuestro componente podemos pasar un string o unarray de string.

Exact: Solo vamos a mostrar nuestro componente cuando la ruta sea exacta. Ej: /home === /home.
Strict: Solo vamos a mostrar nuestro componente si al final de la ruta tiene un slash. Ej: /home/ === /home/
Sensitive: Si le pasamos true vamos a tener en cuenta las mayúsculas y las minúsculas de nuestras rutas. Ej: /Home === /Home

Component: Le pasamos un componente para renderizar solo cuando la ubicación coincide. Eneste caso el componente se monta y se desmonta no se actualiza.

Render: Le pasamos una función para montar el componente en línea.

Instalación

Por defecto, React no viene con un sistema integrado de navegación, ya que debemantener sus dependencias al mínimo porque no todos los proyectos necesitan routing, la misma se maneja con una dependencia aparte.

Tenemos varias formar de instalar, pero de las cuales utilizaremos el siguientecomando:

npm install react-router-dom

Importar

Una vez instalado debemos importar el modulo

```
import React from 'react';
import { BrowserRouter, Switch, Route } from 'react-router-dom';
```

Una vez realizada la importación se necesita realizar varias cosas:1-

Envolver la aplicación en un BrowserRouter

2-Crear un Switch, en el cual se proyectarán las vistas navegadas 3-

Crear las rutas (Route´s) de las distintas secciones para navegar

Configuración del Router

Comencemos a configurar nuestro Router, para eso genera el siguiente archivo dentrode la carpeta ***src/Router.js*** este archivo sustituirá al archivo App.js en el index.js de nuestra aplicación, en él haremos la configuración y la gestión de nuestras rutas:

Primero si queremos proveer a toda nuestra aplicación de rutas tenemos que posicionar al componente ***BrowserRouter*** como el componente padre de todas nuestras rutas es decir este debe envolverlas de la siguiente manera:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
const Router = () => {
 const Home = () => <h1>Home</h1>; const
 Pets = () => <h1>Pet List</h1>; const Layout
 = () => <h1>Layout</h1>;return (
  <>
   <BrowserRouter>
    <Routes>
     <Route path="/" element={<Layout />}></Route>
    </Routes>
   </BrowserRouter>
  </>
 );
};
export default Router;
```

Como se habrá notado agregamos el componente Routes antes de nuestra primera ruta, esto se debe a que Routes genera un árbol de rutas a partir de su props.childrenes decir que cada Route es un elemento hijo de Routes y esto es así, ya que en el momento en que cambiamos la URL en nuestra aplicación, Routes busca en todos suschildren Routes para encontrar la mejor coincidencia y renderizar esa rama en la interfaz del usuario.

```
//src > Router.js
import { BrowserRouter, Routes, Route } from "react-router-dom";const
Router = () => {
 const Home = () => <h1>Home</h1>; const
 Pets = () => <h1>Pet List</h1>; const Layout
 = () => <h1>Layout</h1>;return (
  <>
   <BrowserRouter>
    <Routes>
     <Route path="/" element={<Layout />}></Route>
     <Route path="home" element={<Home />}></Route>
     <Route path="pets" element={<Pets />}></Route>
    </Routes>
   </BrowserRouter>
  </>
 );
};
export default Router;
```

Dentro de nuestro Router podemos generar todas las rutas que queramos solo necesitamos hacer uso del componente Route agregarle al menos dos propiedadespath y element, de esta manera si el path coincide con la URL del navegador renderizara el valor de element.

Perfecto hasta este punto ya tenemos nuestro Router con una configuración básica, podemos dirigirnos a cada una de las rutas que configuramos y obtendremos una vista especifica. Continuemos con el desarrollo de nuestro componente Layout, tener una navbar no solo nos facilitará el desarrollo también provee a nuestros usuarios de los enlaces disponibles dentro de nuestra aplicación, para ello React Router nos proporciona el siguiente componente:

<**Link** /> Este componente recibe en su propiedad to el path de la ruta a la cual nos queremos dirigir dentro de nuestra aplicación. Es importante hacer uso de este, yaque además de permitirnos crear los enlaces de navegación este no recarga nuestra aplicación, característica fundamental de una SPA.

```
import { Link } from "react-router-dom";const

Layout = () => {

 return (

  <>

   <nav style={{ display: "flex", justifyContent: "space-around" }}>

    <Link to="/home">Home</Link>

    <Link to="/pets">Pets</Link>

   </nav>

  </>

 );

};

export default Layout;
```

Rutas Anidadas

Las rutas anidadas nos permiten tener persistencia de componentes es decir nosotros necesitamos que la navbar persista en todas nuestras vistas para lograr esto, la ruta que

renderiza el componente Layout debe envolver a todas las rutas restantes de la siguiente manera:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

import Layout from "./components/Layout";

const Router = () => {

 const Home = () => <h1>Home</h1>;

 const Pets = () => <h1>Pet List</h1>;

 // const Layout = () => <h1>Layout</h1>;

 return (

  <>
   <BrowserRouter>

    <Routes>

     <Route path="/" element={<Layout />}>

      <Route path="home" element={<Home />}></Route>

      <Route path="pets" element={<Pets />}></Route>

     </Route>
    </Routes>

   </BrowserRouter>
  </>

 );

};

export default Router;
```

De esta forma Route Layout paso a ser el componente padre de todas las rutas restantes, puedes imaginas esto como bloques de lego, paso a ser la base de las demás rutas, y podemosir agregando bloques a esta base tanto como queramos, es decir que nuestras rutas hijas podrían seguir anidando y persistiendo en la interfaz del usuario.

Es importante resaltar que este componente Layout no sabe en donde debe renderizar el componente del Route hijo que corresponda con la URL actual, para esto React Router nos proporciona el componente:

Este componente solamente renderiza en el componente padre la siguiente coincidencia en las rutas.

```
import { Link, Outlet } from "react-router-

dom";const Layout = () => {

 return (

  <>

   <nav style={{ display: "flex", justifyContent: "space-around" }}>

    <Link to="/home">Home</Link>

    <Link to="/pets">Pets</Link>

   </nav>

   <Outlet />

  </>

 );

};

export default Layout;
```

Ruta index

Seguro notaste que podemos dirigirnos a la ruta “/home“ y a la ruta “/pets” sin embargo, mientras permanecemos en la ruta raíz “/” Outlet no renderiza nada, podemos agregar unaruta hija que se renderice por defecto de la siguiente manera:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

import Layout from "./components/Layout";

const Router = () => {

 const Home = () => <h1>Home</h1>;
```

```
const Pets = () => <h1>Pet List</h1>;

// const Layout = () => <h1>Layout</h1>;

return (
 <>

  <BrowserRouter>

   <Routes>
    <Route path="/" element={<Layout />}>
     **<Route index element={<Home />}></Route>**

     <Route path="pets" element={<Pets />}></Route>
    </Route>

   </Routes>
  </BrowserRouter>

 </>
);
};

export default Router;
```

De esta forma cuando estemos posicionados en la ruta “/” Outlet tomara el valor del elementde la ruta index para renderizarlo en el componente padre.

Ruta 404

Ahora imagina que nuestro usuario ingresa una URL desconocida… Lo que sucederá es que no se renderizara nada, ya que no existe ninguna ruta correspondiente, podemos configurar unaruta global que se muestre en caso de no haber coincidencias de la siguiente manera:

```
import { BrowserRouter, Routes, Route } from "react-router-dom";

import Layout from "./components/Layout";

const Router = () => {
```

```
const Home = () => <h1>Home</h1>;

const Pets = () => <h1>Pet List</h1>;

// const Layout = () => <h1>Layout</h1>;

return (

 <>
  <BrowserRouter>

   <Routes>
    <Route path="/" element={<Layout />}>

     <Route index element={<Home />}></Route>
      <Route path="pets" element={<Pets />}></Route>

      **<Route path="*" element={<h1>404</h1>}></Route>**

    </Route>
   </Routes>

  </BrowserRouter>
 </>

 );
};

export default Router;
```

useParams

El hook useParams, permite acceder un objeto que retiene cada parámetro dinámico ajustado sobre la ruta. Cuándo se especifica una ruta, esta puede contener parámetros de la forma :fooo :bar, referentes a parámetros dinámicos, que adquirirán el valor que establezca el usuario, sobre la url. Por ejemplo, la ruta /clients/client/:id, podrá acceder al parámetro id que contenga el valor dado por el usuario, como, /clients/client/c018, referente al usuario c018, o si se accede a la ruta /clients/client/d970, ahora el parámetro id tomará el valor de d970. Estosparámetros son útiles para recuperar valores colocados sobre el pathname de la ruta, de tal forma que nos genera una ruta dinámica y rica en información.

Por ejemplo, la ruta /facturas/:uuid/proveedores/:pid, nos permitirá acceder a dos parámetros dinámicos sobre la url, que son uuid y pid, de tal modo, que podremos

consultar los datos del proveedor de una factura, conociendo el uuid de la factura y el id del proveedor.

```
import { useParams } from "react-router-
dom";function Demo() {
 const params = useParams();
 return <span>ID: {params.id}</span>;
}
// Definimos la ruta en Router > Switch
<Route path={{ pathname: "/demo/:id" }}>
 <Demo />
</Route
// Navegamos a la ruta con Link
<Link to="/demo/1">Demo 1</Link>
<Link to="/demo/ABC">Demo ABC</Link>
```

Donde params es un objeto que tendrá como atributos, cada parámetro sobre la ruta especificada mediante :, como en /demo/:id, tendremos { id: "1" } o { id: "ABC" }, según sea elcaso.