educación IT
CODE YOUR FUTURE

ReactJS Developer

Módulo 4

educación IT
CODE YOUR FUTURE

Introducción a *React Router*

React Router

React Router es una completa **librería para implementar enrutamiento del lado del cliente y del lado del servidor para aplicaciones React.**

Esta librería nos servirá para **implementar enrutamiento** o, en otras palabras, **renderizar ciertos componentes en función de eventos en la *URL*** (como el cambio en la url o en alguno de sus parámetros).

Cuándo puedes usarla

- ¿Quieres **renderizar un componente si hay una *URL* específica**?
  Entonces te sirve *React-Router.*
- ¿Quieres **actualizar un componente en función de una *query* de la *URL***?
  Entonces te sirve *React-Router.*
- ¿Quieres **hacer zonas de acceso restringido en tu aplicación**?
  Entonces te sirve *React-Router.*

SPA Routing

React-Router es una capa de conexión (un adaptador) de las funcionalidades de enrutamiento para React.

El enrutamiento del lado del cliente (o *Single Page Application Routing*) consiste en **mostrar determinada interfaz en función de la *URL* sin que haya solicitudes a un servidor,** sino de forma **dinámica** mediante JavaScript.

Para lograr esto, JavaScript ofrece las siguientes posibilidades de forma nativa:

- Usar **el hash (#) de la *URL*** y, en base a eso, **definir la interfaz a visualizar.**
- Usar la ***API History*** que permite **gestionar las *URL* sin necesidad de usar el hash,** de una forma más limpia.

Instalación

Paso a paso

1. Crea un **nuevo proyecto React o escoge uno ya existente.**
2. Abre la línea de comandos en ese proyecto para instalar **React-Router por npm.**
3. Ejecuta **el siguiente comando:**

```
npm install react-router-dom
```

4. Existen varios adaptadores en React-Router. **El adaptador al *DOM* sirve para aplicaciones SPA.**

Componentes principales

Componentes de React Router

Vamos a agrupar los componentes ofrecidos por *React Router* en las siguientes categorías:

- **Routers:** sirven para determinar **el medio que vamos a usar para el enrutamiento** (`hash` / `history` / etc.).
- **Diccionarios de rutas:** componentes que sirven para **definir el diccionario de rutas de la aplicación.**
- **Navegación:** componentes que sirven para **navegar a una ruta específica.**

BrowserRouter

Este router es el medio que se recomienda para la navegación en ***SPA* con React Router.**

El `BrowserRouter` **almacena la información de la ruta actual en memoria, proporcionando una *URL* limpia.** Sirve para la mayoría de los casos. Es el router que usamos **por defecto.**

Sin embargo, hay que tener en cuenta que, al no usar `hash` ni cualquier otro añadido a la *URL*, es necesario contar con un servidor que contemple el **`fallback`** (lo veremos más adelante en esta clase). Resuelto eso, el `BrowserRouter` es la mejor alternativa.

Paso a paso

1. Los `Routers` deben estar en el nivel **más alto del árbol de componentes,** por lo que un buen lugar es el archivo ***index.js***.
2. Vamos al ***index.js***.
3. Importa **`BrowserRouter`** del paquete **`react-router-dom`**.
4. Colocá el componente **`BrowserRouter`** envolviendo toda tu aplicación.

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3  import App from "./App";
4  import { BrowserRouter } from "react-router-dom";
5
6  ReactDOM.render(
7      <BrowserRouter>
8          <React.StrictMode>
9              <App />
10         </React.StrictMode>
11     </BrowserRouter>,
12     document.getElementById("root")
13 );
14
```

HashRouter

`HashRouter` nos va a permitir implementar un enrutamiento en el cual **el estado de la página se almacena en la *URL*.**

Si bien esto conlleva *URLs* menos limpias, también implica que nuestra aplicación React será **más portable** al no necesitar de un servidor que procese las rutas, como es el caso del `BrowserRouter`.

`HashRouter` se usa **cuando el `BrowserRouter` no es viable.**

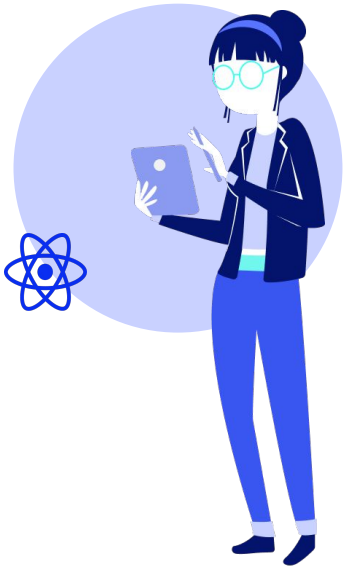
Paso a paso

1. Los `Routers` deben estar en el nivel **más alto del árbol de componentes,** por lo que un buen lugar es el archivo ***index.js*.**
2. Vamos al ***index.js*.**
3. Importa **`HashRouter`** del paquete **`react-router-dom`.**
4. Colocá el componente **`HashRouter`** envolviendo toda tu aplicación.

```
1  import React from "react";
2  import ReactDOM from "react-dom";
3  import App from "./App";
4  import { HashRouter } from "react-router-dom";
5
6  ReactDOM.render(
7      <HashRouter>
8          <React.StrictMode>
9              <App />
10         </React.StrictMode>
11     </HashRouter>,
12     document.getElementById("root")
13 );
14
```

Diccionarios de rutas

Los diccionarios de rutas son **componentes que sirven para definir el diccionario de rutas de la aplicación.**



Routes y Route

Los componentes **`Routes`** y **`Route`** son la principal fuente para renderizar en `React-Router` basado en el valor actual de la propiedad **`location`**.

Piensa en un **`Route`** como una especie de condicional **`IF`**: si su propiedad `path` coincide con la *URL* actual, **entonces renderiza en ese lugar el `element` asociado.**

Un `Route` puede tener a su vez componentes hijos y de esa forma **definir rutas anidadas.**

El componente **`Routes`** sirve para **agrupar un conjunto de rutas.**

Paso a paso

1. Importa **`Routes` y `Route` de `react-router-dom`.**
2. Escribe el componente **`Routes`** en el lugar donde quieres que **se renderice el componente según la *URL* actual.**
3. Como nodos hijos a `Routes`, escribe **un `Route` por cada posible ruta de la aplicación.**
4. Recuerda que cada `Route` debe tener **una propiedad `path`,** que indica la *URL* que busca, y **`element`,** que contiene *JSX* con el componente a renderizar en caso de que la *URL* actual coincida con el valor de su `path`.
5. Puedes crear rutas anidadas poniendo **un `Route` como hijo de otro.**
6. Puedes definir parámetros por la *URL* con la sintaxis **`:miParametro`.** Estos parámetros podrás leerlos en el componente renderizado con el **hook `useParams`.**

```
1   import { Route, Routes } from "react-router-dom";
2
3   function App(props)
4   {
5       return (
6           <Routes>
7               <Route path="/" element={<div>Home Page</div>}>
8
9                   <Route path="contacto" element={<div>Estas en contacto</div>} />
10
11                  <Route path="producto/:id" element={<Producto />} />
12
13              </Route>
14          </Routes>
15      );
16  }
17
18  export default App;
```

Navegación

Los componentes de navegación permiten **navegar a una ruta específica.**

Link

`Link` es un componente que sirve para navegar **a otra página.** Este **renderiza un elemento `<a>`** y lleva a que, al hacer clic izquierdo (o tap en pantallas chicas), **navegue a la *URL* indicada.**

Paso a paso

1. Importa Link de **`react-router-dom`.**
2. Escribe el ***JSX*** **`<Link></Link>`** donde quieras renderizar la etiqueta **`<a>`** para la navegación.
3. Piensa en el `Link` como un **`<a>`** normal. El contenido que ingreses será lo que irá **dentro de la apertura y cierre de dicha etiqueta `<a>`.**
4. Puedes agregarle otras propiedades como **`classList`,** si así lo deseas.
5. Para indicar la ruta, usa la propiedad **`to`**. El valor de esta propiedad será lo que se escriba finalmente en la *URL*.

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

function Producto(props)
{
    return (
        <div>
            <Link to="/contacto">Contacto</Link>
            <Link to={`/detalle/${props.id}`}>Detalle</Link>
        </div>
    );
}

export default Producto;

```

Navigate

Cuando se renderiza un `Navigate`, éste cambia el **`location` actual.** En otras palabras: al renderizarse `Navigate`, provoca **una redirección.**

Lo podemos utilizar para proteger el acceso a páginas renderizando un `Navigate` a una página de `Login` si, por ejemplo, el usuario no está autorizado para la página.

Otro ejemplo, muy usado en los home bankings, es redirigir a la página de inicio de sesión luego de un tiempo de inactividad.

Paso a paso

1. **Importa `Navigate` del paquete `react-router-dom`.**
2. Define la condición que, al cumplirse, **ocasione la redirección.**
3. Haz **un condicional en *JSX*** según lo visto anteriormente.
4. Renderiza el `Navigate` **si esa condición se cumple.**
5. Asigna a la propiedad `to` de `Navigate` **la *URL* que se debe cargar.**

La propiedad **`to`** funciona exactamente igual que en el componente `Link`: es el texto que se escribirá en la *URL*.

```
1   import { Navigate } from "react-router-dom";
2
3   function Dashboard(props)
4   {
5       const isLoggedIn = props.user !== null;
6
7       return (
8           <div>
9               {!isLoggedIn && <Navigate to="/login" />}
10          </div>
11      );
12  }
13
14  export default Dashboard;
15
```

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!