React

Router

Enrique Barra Arias
Álvaro Alonso González

Routing & Navigation

- *El Router* permite la navegación por una SPA

- La navegación ocurre cuando el usuario
  - Introduce una URL en la barra de direcciones
  - Cliquea un link de la página
  - Cliquea los botones de atrás/adelante del navegador

- Gracias al router podemos
  - Manejar la interacción del usuario
  - Modificar la vista que se muestra
  - Acceder a parámetros de la ruta (/posts/15 o /posts/15?search=hola)
  - Definir y anidar rutas (/posts/15/comments/11)
  - Manejar el histórico

- Info y doc oficial: https://reactrouter.com/docs/en/v6

Instalación

```
npm install react-router-dom@6
```

- Ejemplos míos: https://github.com/REACT-UPM/ejemplos_simples_rutas

React router API con ejemplos

- **Router component** <BrowserRouter> y <HashRouter>:
  - <HashRouter> es un <Router> que usa la parte de **hash** (#) de la URL (i.e. window.location.hash) para sincronizar la vista con la URL
  - <BrowserRouter> es un <Router> que usa la **HTML5 history API** (pushState, replaceState y el evento popstate) para sincronizar la vista con la URL

- **Link components** <Link> y <NavLink>
  - Proveen navegación declarativa y accesible en la aplicación
  - <NavLink> es una versión especial de <Link> que añade un estilo determinado al elemento renderizado cuando corresponde con el link de la URL actual
  - Atributos: **to** (página a la que apunta), …

- **Route component** <Route>
  - Su responsabilidad más básica es mostrar una interfaz de usuario determinada cuando la URL actual coincide con el path de la ruta.
  - Attributes: **path**, **element** …

Comenzando

El primer paso es conectar nuestra app a la URL usando BrowserRouter

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import {BrowserRouter} from "react-router-dom";
import LandingRutas2 from "./LandingRutas2";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <LandingRutas2 />
  </BrowserRouter>
);
```

Definiendo rutas

- Necesitamos encerrar todas nuestras rutas (componente **Route**) dentro de un componente **Routes**.
- A través de la prop **path** de cada **Route**, establecemos la URL relativa a través de la cual queremos hacer accesibles nuestros componentes.

```
export default function LandingRutas(props){


  return (<div>
      <h2>Mis primeras rutas</h2>
      <nav style={{borderBottom: "solid 1px", paddingBottom: "1rem"}}>
        <Link to="/">Home</Link>|{" "}
        <Link to="/unapagina">Ver página 1</Link> |{" "}
        <Link to="/otra">Ver página 2</Link>
      </nav>
      <Routes>
        <Route path="/" element={<div>Página principal</div>} />
        <Route path="/unapagina" element={<Pagina />} />
        <Route path="/otra" element={<OtraPagina />} />
      </Routes>
    </div>);
}
```

Navegación

- En lugar de navegar usando la etiqueta HTML <a/>, usamos el componente Link de la librería react-router

```
export default function LandingRutas(props){


  return (<div>
      <h2>Mis primeras rutas</h2>
      <nav style={{borderBottom: "solid 1px", paddingBottom: "1rem"}}>
        <Link to="/">Home</Link>|{" "}
        <Link to="/unapagina">Ver página 1</Link> |{" "}
        <Link to="/otra">Ver página 2</Link>
      </nav>
      <Routes>
        <Route path="/" element={<div>Página principal</div>} />
        <Route path="/unapagina" element={<Pagina />} />
        <Route path="/otra" element={<OtraPagina />} />
      </Routes>
    </div>);
}
```

Router params

Podemos usar rutas con parámetros variables

```
<Route path="/users/:userId" element={<User users={users}/>}  />
```

Podemos hacer uso de esos parámetros para renderizar nuestros componentes

```
function User(props) {
   let {userId} = useParams();
   return <UserProfile user={props.users[userId]}/>
}
```

Navegación

- Utilidad para navegar programáticamente (dar la orden de ir a ruta)
- Si el usuario hace click en un Link esto no hace falta, lo hace el router solo, esto es para desde un método mandar al usuario a una URL
- Hook **useNavigate** que devuelve el método "**navigate**"

```
export default function Invoice() {
  let navigate = useNavigate();

  return (
    <main style={{ padding: "1rem" }}>
      <h2>Total Due: {invoice.amount}</h2>
      <p>Due Date: {invoice.due}</p>
        <button
          onClick={() => {
            deleteInvoice(invoice.number);
            navigate("/invoices");
          }}>
          Delete
        </button>
    </main>
  );
}
```

Repositorios

- https://github.com/REACT-UPM/ejemplos_simples_rutas
- https://github.com/REACT-UPM/ejemplo_completo_rutas
- https://github.com/REACT-UPM/ejemplo_tres_en_raya/tree/router

Mucho más

- Rutas anidadas (**entender especialmente el componente "outlet"**):
  - https://reactrouter.com/docs/en/v6/getting-started/tutorial#nested-routes
- Ruta "No match" (estilo 404):
  - https://reactrouter.com/docs/en/v6/getting-started/tutorial#adding-a-no-match-route
- Search params:
  - https://reactrouter.com/docs/en/v6/getting-started/tutorial#search-params
- Link activo:
  - https://reactrouter.com/docs/en/v6/getting-started/tutorial#active-links
- Como hace el match, rutas layout, rutas índice, rutas compuestas,…:
  - https://reactrouter.com/docs/en/v6/getting-started/concepts