UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Inicial
Front End Developer
ReactJS Manejo de rutas

Universidad Tecnológica Nacional - Facultad Regional Buenos Aires
Medrano 951 (C1179AAQ) C.A.B.A. Tel: (54 11) 4867-7511

Bloques temáticos:

- React router
- Configurar ruteo en app.js
- Configurar enlaces de dirección
- Recibir parámetros por URL
- useHistory
- NavLink
- Redirect
- Not Found

React router

El componente router no es un módulo oficial de React. Nos permitirá hacer una web navegable, es decir poder redirigir entre distintas páginas.

Instalación

Debemos ubicarnos dentro del directorio en el cual está nuestra aplicación y ejecutar:

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

Configurar ruteo en app.js

En el archivo **app.js** debemos realizar lo siguiente:

```
import HomePage from ". /Pages/HomePage"
import RegistroPage from './Pages/RegistroPage';
import Menu from ' ./Components/Menu';
import {
    BrowserRouter as Router,
    Switch,
    Route
} from "react-router-dom";
function App() {
    return (
        <div className="App">
            <Router>
                <Menu />
                <Switch>
                    <Route path="/registro" >
                        <RegistroPage />
                    </Route>
                    <Route path="/">
                        <HomePage />
                    </Route>
                </Switch>
            </Router>
        </div >
    );
}
export default App;
```

Como vemos debemos incluir los módulos `Route`, `Router (BrowserRouter)` y `Switch`

Si ejecutamos nuestra aplicación ingresara al contenido del componente App y si en la barra de direcciones colocamos `/` ingresara al contenido del componente `HomePage`

Si en la barra de direcciones colocamos `/registro` entonces se visualizará el contenido del componente `RegistroPage`

Vemos que el elemento `route` recibe 1 propiedad `path` (indica la url por la que se ingresara a ese componente).

Configurar enlaces de dirección

Cuando queramos armar los enlaces (lo que sería `<a href`) lo debemos hacer usando el componente `Link`:

```
import React from "react"
import {
    Link
} from "react-router-dom"
function Menu() {
    return (
        <div>
            <ul> <li>
                <Link to="/">Home</Link>
            </li>
                <li>

                    <Link to="/registro">Registro</Link>
                </li>
            </ul>
        </div>
    )
}
export default Menu;
```

En este ejemplo vemos el componente `Link` aplicado al menú, pero podemos aplicarlo a cualquier componente.

En la propiedad `to` colocamos la url destino (sería el href del componente Link)

Recibir parámetros por URL

En caso de querer recibir parámetros por URL vamos a indicar en el componente route lo siguiente:

```
<Route path="/detalle/:id" >
    <DetallePage />
</Route>
```

En caso de ser un componente de tipo clase este elemento `:id` se enviará al componente `DetallePage` como una propiedad, de la siguiente manera:

```
constructor(props){
    super(props)
    console.log(this.props.params.id)
}
```

En caso de ser un componente tipo función podemos utilizar el hook `useParams()` de la siguiente manera:

```
import React from "react"
import {
    useParams
} from "react-router-dom";
function DetallePage() {
    const { id } = useParams()
    console.log(id)
    return (
        <div>DetallePage</div>
    )
}
export default DetallePage;
```

En ambos casos se hace referencia al elemento `id` porque se identificó el parámetro de esa manera en el route (:id)

useHistory

Hook que permite acceder al histórico de navegación. Podemos utilizar para realizar una redirección `(push)` desde la lógica del componente. Por ejemplo:

```
import React from "react"
import {
    useHistory
} from "react-router-dom";
function RegistroPage() {
    let history = useHistory();
    function handleClick() {
        history.push("/home");
    }
    return (
        <div>
            Registro
            <button type="button" onClick={handleClick}>
                Go home
            </button>
        </div>
    )
}
export default RegistroPage:
```

En este caso al hacer click en **Go To Home** llamamos a la función `handleClick` y en dicha función hacemos un `history.push("/home")` el cual nos redirigirá a la página de inicio.

NavLink

Este componente tiene una serie de propiedades utilizadas a la hora de armar un `navbar` en nuestra aplicación. Ejemplo de utilización en el Menú:

```
function Menu() {
    return (
        <div>
            <NavLink to="/" exact >Home</NavLink>
            <NavLink to="/registro">Registro</NavLink>
        </div>)
}
export default Menu;
```

Se puede ver la totalidad de las propiedades en https://reactrouter.com/web/api/NavLink

Redirect

Permite realizar redirecciones en la lógica de ruteo. Lo aplicamos en el Switch en el cual se realiza la comparación de las direcciones:

```
<Redirect from=" /home" to="/" />
<Route path="/">
<HomePage />
</Route>
<Route path="/" >
<HomePage />
</Route>
```

En el ejemplo cuando recibamos `/home` redirigirá a `/`

https://reactrouter.com/web/api/Redirect

Not Found

Para hacer la lógica de ruteo de nuestra página `404 Not Found` debemos declarar la siguiente regla:

```
<Router>
    <Menu />
    <Switch>
        <Route path="/" exact >
            <HomePage />
        </Route>
        <Route path="/registro" exact >
            <RegistroPage />
        </Route>
        <Route path="/detalle/:id" exact >
            <DetallePage />
        </Route>
        <Redirect from="/home" to="/" exact />
        <Route path="*">
            <NotFoundPage />
        </Route>
```

```
    </Switch>
</Router>
```

De esta manera estaremos accediendo a cualquier url no identificada en reglas anteriores.

Bibliografía y Webgrafía utilizada y sugerida

Fedosejev, A. (2015). React.js Essentials (1 ed.). EEUU, Packt.

Amler, . (2016). ReactJS by Example (1 ed.). EEUU, Packt.

Stein, J. (2016). ReactJS Cookbook (1 ed.). EEUU, Packt.

https://medium.com/@pshrmn/a-simple-react-router-v4-tutorial-7f23ff27adf

https://reactrouter.com/web/guides/quick-start