Jugando con React Router DOM

DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Juguemos

- Vamos a setear nuestro proyecto instalando React Router DOM y dejando el código limpio.
- Dejamos nuestra app como el padre de las rutas.
- Vemos que tenemos dos rutas: una principal que sería newGame y, luego, una dinámica que espera un género y renderiza Character.

Aquí el código:

```
import React, { Component } from 'react'
import { BrowserRouter, Route } from 'react-router-dom'
import NewGame from './NewGame'
import Character from './Character'

export default class App extends Component {
 render() {
   return (
     <BrowserRouter>
       <Route exact path="/" component={NewGame}/>
       <Route exact path="/character/:genre" component={Character}/>
     </BrowserRouter>
   )
 }
}
```

Podemos ver que en NewGame estamos utilizando un link donde le pasamos el tipo de género que tendrá nuestro personaje.

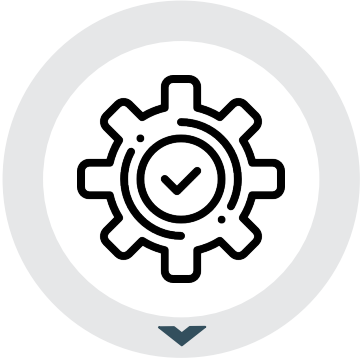
```
import React, { Component } from 'react'
import { Link } from 'react-router-dom'
import "./styles.css"

export default class NewGame extends Component {
   render() {
       return (
           <div className="container">
               <h1>Vamos a crear a nuestro personaje con react router dom</h1>
               <h3>¿Genero de tu personaje?</h3>
               <div className="buttons">
                   <Link to="/character/hombre">HOMBRE</Link>
                   <Link to="/character/mujer">MUJER</Link>
               </div>
           </div>
       )}}
```

Mirá una página deployada acá.

Podemos observar que...

Recibimos algo por props llamado match. Esto nos devuelve los parámetros o URL como datos que nos pueden servir.



Usamos el URL como parámetro y, al final, agregamos un botón para volver al inicio.

Aquí el código:

```
import React, { Component } from 'react'
import { Link } from 'react-router-dom'
import "./styles.css";
export default class Character extends Component {
   render() {
       const genero = this.props.match.params.genero;
       const url = this.props.match.url
       return (
           <div className="container">
               <h1>{`El genero de tu personaje es${genero}`}</h1>
               <img width="240px" src={`/images/${genero}.png`} alt={`${genero}`} />
               <h3>¿Clase de tu personaje?</h3>
               <div className="buttons">
                   <Link to={`${url}/warrior`}>{`Guerrer${genero === "hombre" ? "o" : "a"}`}</Link>
                   <Link to={`${url}/archer`}>{`Arquer${genero === "hombre" ? "o" : "a"}`}</Link>
                   <Link to={`${url}/mage`}>{`Mag${genero === "hombre" ? "o" : "a"}`}</Link>
               </div>
               <Link id="gameButton" to="/">Volver</Link>
           </div>
       )
   }}
```

¿Qué pasa si el usuario escribe una ruta inexistente?

```
import React, { Component } from 'react'
import "./styles.css"
export default class NotFound extends Component {
   render() {
       return (
           <div className="container">
               <h1>NOT FOUND</h1>
           </div>
       )
   }
}
```

Para este caso, vamos a hacer dos cosas:

1) Crear un componente que se renderizará para todas las rutas que no lleven a nada.
2) Agregar algo a nuestra App.jsx

¿Qué pasa si el usuario escribe una ruta inexistente?

Agregamos algo llamado **Switch**

- Si no lo utilizamos, la ruta que utiliza el asterisco, se renderizará siempre (solo queremos que ocurra cuando NO matchea con las nuestras).
- El Switch nos permite renderizar de a un componente por vez.

Aquí el código

```
import React, { Component } from 'react'
import { BrowserRouter, Route, Switch } from 'react-router-dom'
import NewGame from './NewGame'
import Character from './Character'
import NotFound from './NotFound'

export default class App extends Component {
 render() {
   return (
     <BrowserRouter>
       <Switch>
         <Route exact path="/" component={NewGame} />
         <Route exact path="/character/:genero" component={Character} />
         <Route path="*" component={NotFound} />
       </Switch>
     </BrowserRouter>
   )
 }}
```

Finalizando el juego

Vamos a terminar con un último archivo y agregándolo a nuestra App.jsx.

```
<Switch>
    <Route exact path="/" component={NewGame} />
    <Route exact path="/character/:genero" component={Character} />
    <Route exact path="/character/:genero/:clase" component={EndGame} />
    <Route path="*" component={NotFound} />
</Switch>
```

Agregamos una nueva ruta que tiene un género y una clase dinámica.

Finalizando el juego

En este nuevo archivo, vemos que nos sigue reconociendo ambos parámetros. Además usamos una props llamada history que nos permite volver a la página anterior.

```
import React, { Component } from 'react'
import "./styles.css"

export default class EndGame extends Component {
   render() {
       const {genero, clase} = this.props.match.params;
       const goBack = this.props.history.goBack
       return (
           <div className="container">
               <h1>ENHORABUENA</h1>
               <img width="240px" src={`/images/${clase}.png`} alt={`${clase}`} />
               <h3>{`Tu personaje es ${genero} y su clase es ${clase}.`}</h3>

               <button id="gameButton" onClick={goBack} >Volver</button>
           </div>
       )}}
```

DigitalHouse>