Curso de React Router y Redux

# ¿Qué es React Router y cómo instalarlo?

Si estás empezando la escuela de JavaScript desde este curso deberás crear una copia del repositorio de Platzi Video:

git **clone** **https**://github.com/platzi/PlatziVideo.git

Una vez tienes listo el repositorio vas a crear una nueva rama para trabajar en ella a lo largo del curso:

git checkout -**b** feature/router-redux

Ya que nos encontramos dentro de la rama vamos a instalar React Router, la librería que nos va a permitir manejar rutas dentro de nuestra aplicación:

npm **install** react-router-dom --save

# ¿Qué es React Router y cómo instalarlo?

Dentro de nuestro proyecto vamos a crear una carpeta llamada routes donde vamos a ir añadiendo las rutas que necesitemos en la aplicación.

Las rutas que añadamos debemos definirlas con el componente Route y estas deben estar encapsuladas dentro del componente BrowserRouter del paquete de react-router-dom. Para definir una ruta con el componente Route debemos pasarle las props de:

* path para indicar la url.
* exact si queremos que funcione única y exactamente con la url que le digamos.
* component para indicarle el componente que va a renderizar.

npm run start

json-server initialState.json

Recien tuve un tipo en App.js

**import** React **from** "react";

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

**import** Home **from** "../containers/Home";

**const** App = () => (

<BrowserRouter>

<Route exact path="/" component={Home} />

</BrowserRouter>)

export default App;

**es con parentesis, no con llaves**. Tarde 15 minutos en darme cuenta jaja!

Yo tuve un pequeño problema con el json-server porque no lo tenía instalado globalmente, ejecutando el siguiente comando sirvio perfecto todo lo demás.

npm **install** -g **json**-**server**

Tomado de: <https://www.npmjs.com/package/json-server>

# Container: Login

Vamos a descargar el proyecto del curso de Frontend de escuela de JavaScript para crear los componentes que necesitaremos, en caso de que vengas del curso de React no debes descargar nada.

Debemos modificar nuestra configuración del entorno de desarrollo local para que pueda funcionar con el uso de rutas, debemos ir al archivo webpack.config.js y añadir este fragmento de código antes de plugins:

module.exports = {

{/\*...\*/}

devServer: {

historyApiFallback: **true**,

},

{/\*...\*/}

}

Aqui esta el codigo de section para quien lo necesite

<section className='login'>

<section className='login\_\_container'>

<h2>Inicia sesión</h2>

<form className='login\_\_container--form'>

<input className='input' type='text' placeholder='Correo' />

<input className='input' type='password' placeholder='Contraseña' />

<button className='button'>Iniciar sesión</button>

<div className='login\_\_container--remember-me'>

<label>

<input type='checkbox' id='cbox1' value='first\_checkbox' />

Recuérdame

</label>

<a href='/'>Olvidé mi contraseña</a>

</div>

</form>

<section className='login\_\_container--social-media'>

<div>

<img src='../assets/google-icon.png' /> Inicia sesión con Google

</div>

<div>

<img src='../assets/twitter-icon.png' /> Inicia sesión con Twitter

</div>

</section>

<p className='login\_\_container--register'>

No tienes ninguna cuenta <a href=''>Regístrate</a>

</p>

</section>

</section>

# Container: Register

Para asegurar que nuestras rutas solamente se rendericen con la que haga match con la url debemos encapsular las rutas dentro del componente.

Atributos para los Route Objectos:

* path: la ruta en la que se renderizará el componente en forma de cadena de texto
* exact: un booleano para definir si queremos que la ruta tiene o no que ser exacta para renderizar un componente. Eg: /index !== /index/all.
* strict: un booleano para definir si queremos que el último slash sea tomado en cuenta para renderizar un componente. Eg: /index !== /index/.
* sensitive: un booleano para definir si queremos distinguir entre minúsculas y mayúsculas, y tomar esto en cuenta para renderizar un componente. Eg: /index !== /Index
* component: recibe un componente a renderizar. Crea un nuevo elemento de React cada vez. Esto causa que el componente se monte y desmonte cada vez (no actualiza).
* render: recibe un método que retorna un componente. A diferencia de component no remonta el componente.

La diferencia entre poner el switch y no ponerlo es que cuando tienes el mismo path para todos solamente toma el primero y lo renderiza.

# Container: 404 Not Found

Es importante siempre tener una ruta que renderice un componente para las urls que no existan, debemos añadir esta ruta al final del Switch para que sea el caso por default.

Fragment nos permite no añadir elementos extra al DOM, podemos utilizar Fragment de 2 maneras:

1. Añadiendo el componente o .
2. O implemente encapsulando nuestros elementos dentro de <>.

# Componente: Layout



# Manejando enlaces y configuraciones

El componente es similar a un elemento ya que nos permite navegar dentro de la aplicación, pero sin la necesidad de tener que recargar la página. Para indicarle el destino a simplemente debemos pasarle la prop to='/mi-enlace'.

También es bueno porque no tiene limitaciones para darle parámetros, bueno para el SEO 😄

<Link to="/" rel="canonical">

<img src={Logo} widht="100" />

</Link>

