Tema 5

React-Router y APIs

React-Router

React-Router es una librería potente de enrutamiento que facilita añadir nuevas pantallas y flujos a las aplicaciones rápidamente, manteniendo la URL sincronizada con lo que se muestra en la página.

Tiene diversas características, como cargar código perezoso, enlazar rutas de manera dinámica y manejo de transición de localización.

Para instalar React-Router en nuestro proyecto, deberemos situarnos en la carpeta del mismo y ejecutar el siguiente comando en la consola:

```
$ npm i react-router-dom@next --save-dev
```

React-Router

Una vez creado nuestro proyecto mediante la orden create-react-app y después de haber instalado React-Router en nuestro proyecto, podemos realizar una prueba sencilla para ver su funcionamiento. En la clase App.js, sustituye el código por el siguiente:

```
import React, { Component } from 'react';
import {BrowserRouter as Router, Route} from 'react-router-dom';

const Home = () => <h1>Hola desde Home!</h1>
const Address = () => <h1>Nos encontramos en la Avenida de Valencia.</h1>
class App extends Component {
  render() {
    return (
      <Router>
        <div>
          <Route exact path="/" component ={Home} />
          <Route path="/address" component ={Address} />
        </div>
      </Router>
    )
  }
}

export default App
```

React-Router

Para poder navegar cómodamente entre páginas, se utiliza el componente **<Link>**. <Link> es similar a utilizar una etiqueta HTML de anclaje.

A la hora de utilizarlo, ha de crearse un nuevo componente que contenga componentes <Link>. De esta manera, tendremos algo como esto:

```
import { Link } from 'react-router-dom';
const Links = () => (
  <div>
    <Link to='/'>Home</Link>
    <Link to='/address'>Address</Link>
  </div>
)
```

Para poder hacer el componente persistente a través de todas las páginas, agruparemos todas nuestras rutas en un componente **<Route>** principal. Habrá que actualizar nuestro componente Home y crear un contenedor.

React-Router

Modificaremos el código de nuestro componente App de manera que se añadan los Links definidos anteriormente.

```
class App extends Component {
  render() {
    return (
      <Router>
        <div>
        <Links />
          <Route exact path="/" component ={Home} />
          <Route path="/address" component ={Address} />
        </div>
      </Router>
    )
  }
}
```

React-Router

Para añadir múltiples rutas dentro de las mismas, se debe modificar el componente a renderizar. Por ejemplo, podemos modificar el componente Home para que contenga dos rutas de la siguiente forma:

```
const Home = () =>
  <div>
    <h3>Enlaces:</h3>
    <Link to='/hola'>Hola</Link>
    <h1/>
    <Link to='/mundo'>Mundo</Link>
  </div>
```

React-Router

Se puede aplicar estilos a los Links en base a si la ruta está activa. Para añadir el estilo a un Link se creará una constante nueva:

```
const CustomLink = ({ activeStyle, children, className, to, activeOnlyWhenExact }) => (
  <Route path={to} exact={activeOnlyWhenExact} children={({ match }) => (
    <Link to={to} className={className} style={match && activeStyle}>{children}</Link>
  )}/>
);
```

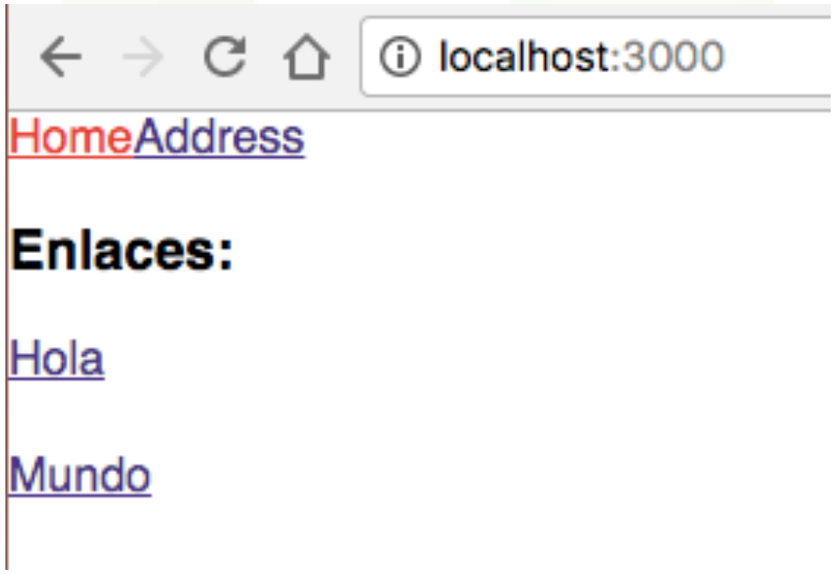
Una vez creado, habrá que modificar la constante Links:

```
const Links = () => (
  <div>
    <CustomLink activeStyle={{color: 'red'}}  to='/'>Home</CustomLink>
    <Link to='/address'>Address</Link>
  </div>
)
```

React-Router

Si ejecutamos la aplicación, notaremos que *Home* siempre está marcada.

Es lógico, ya que todas las demás rutas son sus descendientes.



Promesas

En JavaScript una promesa es un objeto utilizado para computaciones asíncronas. Representa un valor que puede estar disponible ahora, en el futuro o nunca.

Para crear una promesa en React, se realiza de forma similar que en JavaScript:

```
let prom = new Promise(function (resolve, reject) {
 setTimeout(function () {
   resolve('a value')
 }, 100)
})
```

API

Una **API** es un conjunto de funciones, subrutinas y métodos que ofrece una biblioteca para ser utilizado en las aplicaciones. Se suelen utilizar en las bibliotecas de programación.

En React se pueden utilizar APIs de manera similar a JavaScript. Una vez importada la biblioteca correspondiente, se pueden utilizar los métodos o funciones que proporciona. Para importar una biblioteca se utilizan sentencias como la siguiente:

```
import React from 'react'
```

API

Vamos a ver un ejemplo de utilización con react-http-request. Esta librería nos permite utilizar el protocolo HTTP en las aplicaciones React. Se ejecuta el siguiente comando de instalación en la consola:

```
npm install --save react-http-request
```

A continuación utilizaremos esta biblioteca para hacer peticiones a páginas web y conseguir ciertos datos.

API

```
import React, { Component } from 'react'
import { Router, Route, Link, IndexRoute, hashHistory, browserHistory, DefaultRoute,
IndexLink } from 'react-router'
import Request from 'react-http-request';

export default class App extends Component {
  render() {
    return (
      <Request
        url='https://swapi.co/api/planets/1/?format=json'
        method='get'
        accept='application/json'
        verbose={true}
      >
        {
          ({error, result, loading}) => {
            if (loading) {
              return <div>loading...</div>;
            } else {
              var obj = JSON.stringify(result.body)
              var obj2 = JSON.parse(obj)
              console.log(obj2.name)

              return <div>{ JSON.stringify(result.body)
               }</div>;
            }
          }
        }
      </Request>
    );
  }
}
```

API

Analizando el código anterior, vemos que se utiliza la función Request que hemos importado de la biblioteca. Mediante esta función, se pueden realizar peticiones a páginas web y obtener su contenido, el cual se puede filtrar para obtener únicamente el cuerpo del mensaje mediante result.body.

Mediante JSON.parse(obj), podemos acceder a las propiedades del objeto en caso de que sea un diccionario como en este caso.

En resumen, la utilización de APIs es bastante útil e interesante, ya que añade muchísimas funcionalidades nuevas a una aplicación.