Curso de React y React Native

Clase 05

Agenda de la clase

Agenda

- Repaso.
- Routing en React.
- Ejercicios.

Repaso

Routes en React

Routes

El Routing es un aspecto importante a la hora de desarrollar una aplicación web.

Esto se refiere principalmente a las distintas “rutas” de nuestra App, por ejemplo, si existe una sección que sólo se accede con credenciales. Otro caso, en una App del estilo Marketplace tendremos una sección central de navegación y luego una sección de “Ver mi carrito”.

En React esto ha sido implementado de varias formas por varias librerías. Algunas de las más conocidas son:

- React Router
- Redux First Router
- React Navigation (React Native)
- React Mini Router
- Etc.

React Router v5 (1)

Durante el curso se va a estar usando **React Router v5**, que es la librería más comúnmente utilizada. Además, tiene una muy buena documentación.

En React Router, la mayoría de los elementos de navegación son Componentes de React, tratando de mantener el modo de pensar declarativo que se utiliza al desarrollar aplicaciones en React.

Para poder utilizarlo, se necesita instalarlo en el proyecto por `npm` o `yarn`:

- `$ npm install --save 'react-router-dom'`
- `$ yarn add 'react-router-dom'`

React Router v5 (2)

Se van a utilizar algunos de los componentes que brinda la librería para construir el ruteo de la App. Por ejemplo, `BrowserRouter` se utiliza para envolver toda la App y `Route` para declarar **qué rutas** renderizan **qué componentes**.

```
const BasicExample = () => {
   return (
       <BrowserRouter>
           <div>
               <Route exact path="/" component={Home} />
               <Route path="/about" component={About} />
           </div>
       </BrowserRouter>
   );
}
```

React Router v5 (3)

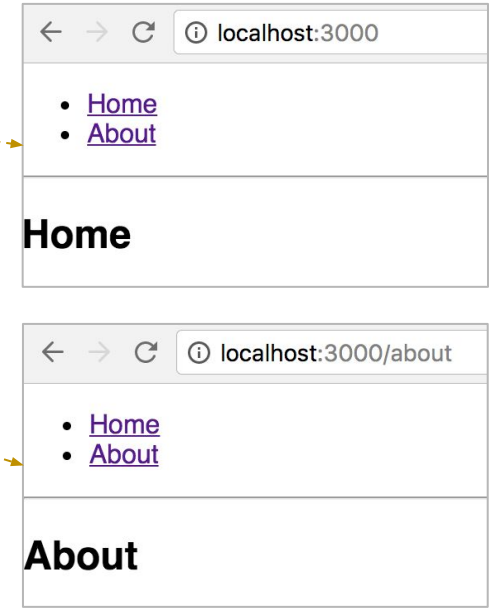
El componente `Link` sirve para declarar navegación a rutas determinadas. Estos componentes hay que importarlos de la librería antes de poder utilizarlos.

```
import { BrowserRouter, Route, Link } from "react-router-dom";
const BasicExample = () =>
   <BrowserRouter>
       <div>
           <ul>
               <li><Link to="/">Home</Link></li>
               <li><Link to="/about">About</Link></li>
           </ul>
           <hr />
           <Route exact path="/" component={Home} />
           <Route path="/about" component={About} />
       </div>
   </BrowserRouter>;
```

`exact` se utiliza para que no "matchee" `Home` en `About`

React Router v5 (4)

```
import { BrowserRouter as Router, Route, Link } from "react-router-dom";
const Home = () =>
    <div>
        <h2>Home</h2>
    </div>;
const About = () =>
    <div>
        <h2>About</h2>
    </div>;
const BasicExample = () =>
    <Router>
        <div>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/about">About</Link></li>
            </ul>
            <Route exact path="/" component={Home} />
            <Route path="/about" component={About} />
        </div>
    </Router>;
```



React Router v5 (5)

El componente `NavLink` es muy similar a `Link` pero permite darle además una propiedad `activeClassName` para que si se está en esa ruta, le aplique la clase de `css` que se le haya pasado.

```
import { BrowserRouter, Route, NavLink } from "react-router-dom";
const BasicExample = () =>
    <BrowserRouter>
        <div>
            <ul>
                <li><NavLink exact activeClassName="active" to="/">Home</NavLink></li>
                <li><NavLink activeClassName="active" to="/about">About</NavLink></li>
            </ul>
            <hr />
            <Route exact path="/" component={Home} />
            <Route path="/about" component={About} />
        </div>
    </BrowserRouter>;
```

React Router v5 (6)

Todos los componentes que son dibujados por `Route` van a recibir en sus Props una propiedad llamada `match`, que tiene acceso a varias cosas de la ruta como la URL.

```
const BasicExample = () =>
 <Router>
   <div>
     <ul>
       <li><Link to="/">Home</Link></li>
       <li><Link to="/topics">Topics</Link></li>
     </ul>
     <hr />
     <Route exact path="/" component={Home} />
     <Route path="/topics" component={Topics} />
   </div>
 </Router>;
```

```
const Topics = (props) =>
   <div>
       <h2>Topics</h2>
       <h3>La url es: {props.match.url}</h3>
   </div>;
```

```
const Topics = ({ match }) =>
   <div>
       <h2>Topics</h2>
       <h3>La url es: {match.url}</h3>
   </div>;
```

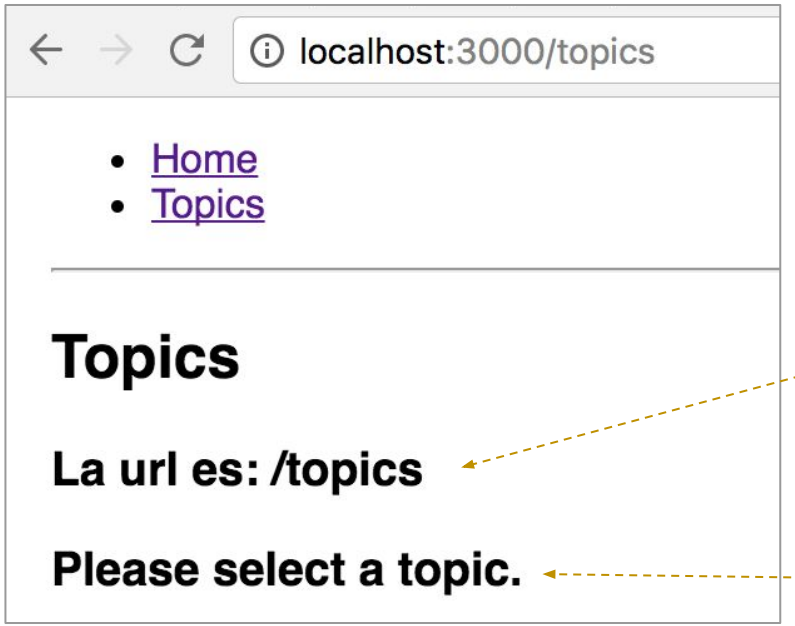
React Router v5 (7)

Cuando `Route` "matchee" la ruta, va a dibujar el componente que reciba en `component`. También se puede usar `render` para indicarle qué renderizar.

```
const BasicExample = () =>
 <Router>
   <div>
     <ul>
       <li><Link to="/">Home</Link></li>
       <li>
          <Link to="/topics">Topics</Link>
       </li>
     </ul>
     <hr />
     <Route exact path="/" component={Home} />
     <Route path="/topics" component={Topics} />
   </div>
 </Router>;
```

```
const Topics = ({ match }) =>
 <div>
   <h2>Topics</h2>
   <h3>La url es: {match.url}</h3>
   <Route exact path={match.url}
     render={
       () => <h3>Select a topic.</h3>
     }
   />
 </div>;
```

React Router v5 (8)



Este es el valor de: {`match.url`}

Esto aparece solo porque hay un `Route` que "matchea" exactamente con {`match.url`} dentro de `Topics`

React Router v5 (9)

Dentro de la prop `match` que reciben los componentes, hay una propiedad `param` donde van a llegar todos los parámetros que se definan en la ruta, con el mismo nombre con el que fueron creados.

```
const Topics = ({ match }) =>
    <div>
        <h2>Topics</h2>
        <Route path={`${match.url}/:topicId`} component={Topic} />
        <Route
            exact path={match.url}
            render={() => <h3>Please select a topic.</h3>}
        />
    </div>;

const Topic = ({ match }) =>
    <div>
        <h3>{match.params.topicId}</h3>
    </div>;
```

React Router v5 (10)

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

const BasicExample = () =>
    <Router>
        <div>
            <ul>
                <li><Link to="/">Home</Link></li>
                <li><Link to="/about">About</Link></li>
                <li><Link to="/topics">Topics</Link></li>
            </ul>
            <hr />
            <Route exact path="/" component={Home} />
            <Route path="/about" component={About} />
            <Route path="/topics" component={Topics} />
        </div>
    </Router>;
export default BasicExample;
```

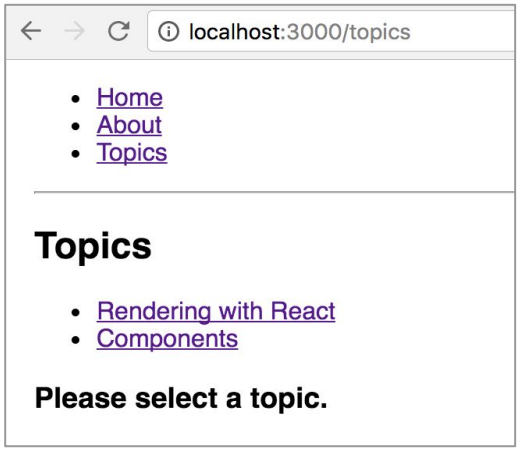
React Router v5 (11)

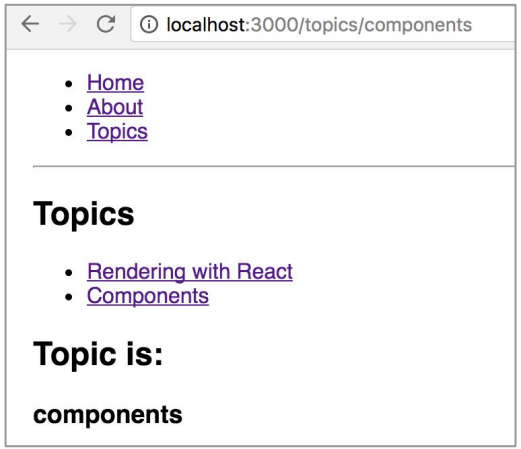
```
const Topics = ({ match }) =>
    <div>
        <h2>Topics</h2>
        <ul>
            <li>
                <Link to={`${match.url}/rendering`}>
                    Rendering with React
                </Link>
            </li>
            <li>
                <Link to={`${match.url}/components`}>
                    Components
                </Link>
            </li>
        </ul>
        <Route path={`${match.url}/:topicId`} component={Topic} />
        <Route exact path={match.url} render={() => <h3>Please select a topic.</h3>} />
 </div>;
```

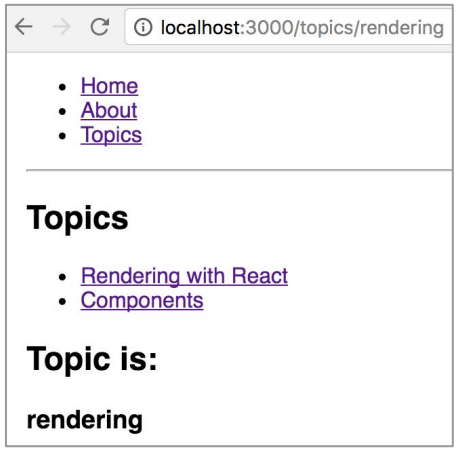
React Router v5 (12)

```
const Home = () =>
    <div>
        <h2>Home</h2>
    </div>;

const About = () =>
    <div>
        <h2>About</h2>
    </div>;

const Topic = ({ match }) =>
    <div>
        <h2>Topic is: </h2>
        <h3>{match.params.topicId}</h3>
    </div>;
```

React Router v5 (13)







React Router v5 (14)

En el caso de tener rutas **ambiguas**, en la que dos pueden matchear, se puede utilizar el `Switch` para envolver las rutas y va a devolver la primera que coincida.

```
const AmbiguousExample = () => <Router>
  <div>
    <ul>
      <li><Link to="/about">About Us (static)</Link></li>
      <li><Link to="/company">Company (static)</Link></li>
      <li><Link to="/kim">Kim (dynamic)</Link></li>
    </ul>
    <Switch>
      <Route path="/about" component={About} />
      <Route path="/company" component={Company} />
      <Route path="/:user" component={User} />
    </Switch>
  </div>
</Router>;
```

En este caso, cuando se navegue a `/about` o `/company` la tercer ruta también "matchea", pero el `Switch` devuelve la primera que coincida.

React Router v5 (15)

```
import React from "react";
import { BrowserRouter as Router, Route, Link, Switch } from "react-router-dom";
const AmbiguousExample = () => <Router>
   <div>
     <ul>
       <li><Link to="/about">About Us (static)</Link></li>
       <li><Link to="/company">Company (static)</Link></li>
       <li><Link to="/kim">Kim (dynamic)</Link></li>
       <li><Link to="/chris">Chris (dynamic)</Link></li>
     </ul>
     <Switch>
       <Route path="/about" component={About} />
       <Route path="/company" component={Company} />
       <Route path="/:user" component={User} />
     </Switch>
   </div>
 </Router>;
const About = () => <h2>About</h2>;
const Company = () => <h2>Company</h2>;
const User = ({ match }) => <h2>User: {match.params.user}</h2>;
```

React Router v5 (16)

Se puede utilizar el componente `Redirect` para indicar cuándo se quiere que determinada ruta cambie a otra automáticamente. Otro detalle importante es que un `Route` sin `path` siempre va a matchear, por eso en el siguiente caso se envuelve en un `Switch`, pero sirve para las rutas no definidas.

```
<Switch>
    <Route path="/" exact component={Home} />
    <Redirect from="/old-match" to="/will-match" />
    <Route path="/will-match" component={WillMatch} />
    <Route component={NoMatch} />
</Switch>
```

React Router v5 (17)

```
import { BrowserRouter as Router, Route,
           Link, Switch,  Redirect } from "react-router-dom";
const NoMatchExample = () =>
<Router>
 <div>
   <ul>
     <li><Link to="/">Home</Link></li>
     <li><Link to="/old-match">Old, to be redirected</Link></li>
     <li><Link to="/will-match">Will Match</Link></li>
     <li><Link to="/will-not-match">Will Not Match</Link></li>
   </ul>
   <Switch>
     <Route path="/" exact component={Home} />
     <Redirect from="/old-match" to="/will-match" />
     <Route path="/will-match" component={WillMatch} />
     <Route component={NoMatch} />
   </Switch>
 </div>
</Router>;
```

```
const Home = () => <p> Home </p>;

const WillMatch = ({ match }) =>
 <div>
   <h2>Match.url is: {match.url}</h2>
   <h3>Matched!</h3>
 </div>;

const NoMatch= ({ match, location }) =>
 <div>
   <h2>Match.url is: {match.url}</h2>
   <h3>No match for
{location.pathname}</h3>
 </div>;
```

Ejercicios

Ejercicios

1. Crear un nuevo proyecto de React.
2. Instalar `react-router-dom`.
3. Crear una homepage (en la ruta “/”) que contenga links para mostrar propuestas laborales en San Francisco, New York y remotos.
4. Al clickear un link, entramos a la ruta de propuestas laborales en la *location* en cuestión, listando los openings disponibles y sus datos.
5. Verificar que la app funcione correcto aunque la navegación no sea “manual”, es decir, entrando con la URL directamente en una pestaña nueva, ej: http://localhost:3000/remote.

Para obtener los datos, utilizaremos la API de GitHub Jobs.