React intermedio

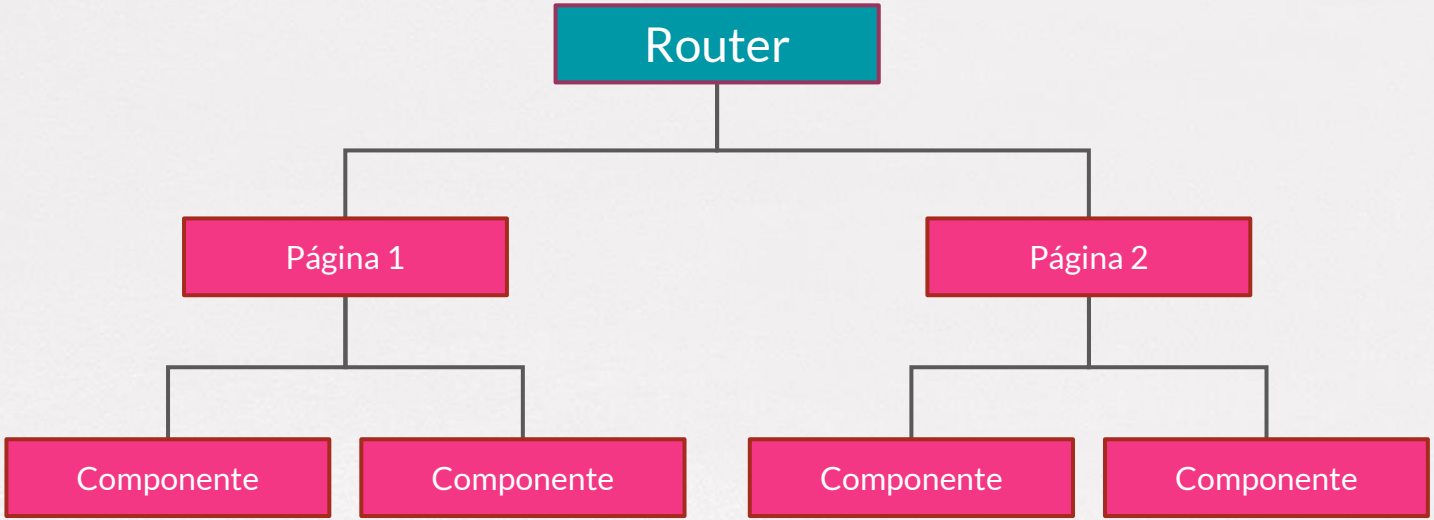
Rutas, navegación y portales

Rutas absolutas y relativas

ÍNDICE

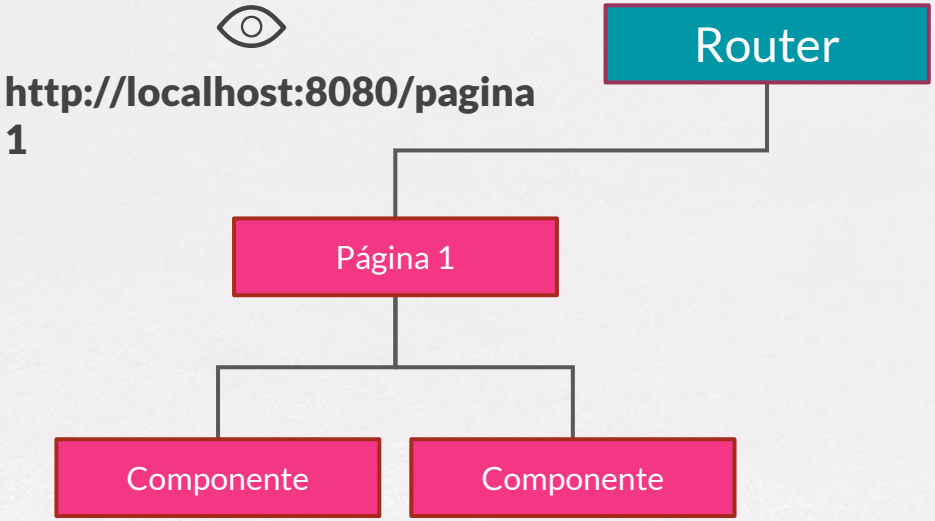
- El enrutamiento en una SPA
- Introduciendo react-router
- Switches y rutas en react-router
- Navegación básica por la aplicación

Enrutamiento en una SPA



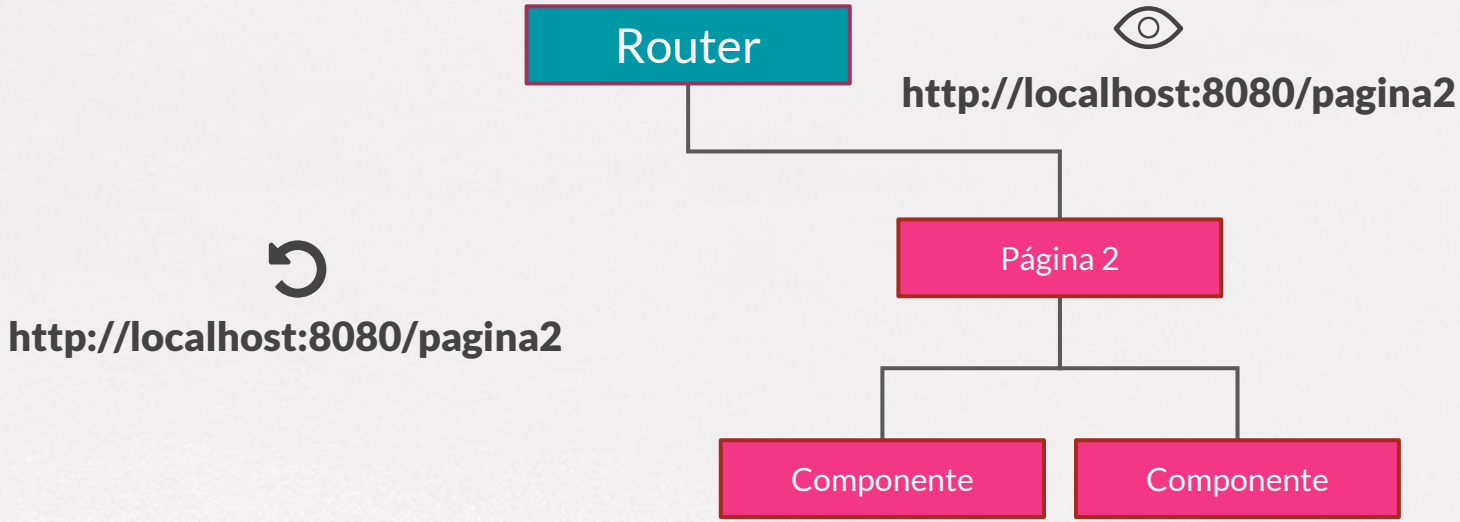
- El router de una SPA es un “falso enrutador”, que realmente no produce la recarga de la página

Enrutamiento en una SPA



- Este componente React sólo actúa como un switch que carga el componente adecuado respecto a la ruta

Enrutamiento en una SPA



- La potencia del enrutamiento es no sólo hacer el switch, si no permitir también la navegación con parámetros y conservar el historial

Introduciendo React router

- React router es una librería de enrutamiento para React con gran popularidad
- Se distribuye en forma básica con la posibilidad de **añadir historial** y **conexión con Redux**
- Realmente la librería a usar es irrelevante, siempre y cuando disponga de las funcionalidades básicas comentadas
- Una curiosidad de React Router es que el mismo equipo desarrolla otro router similar en **React Native**

Introduciendo React router

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter } from "react-router-dom";

function App() {
  return <h1>El ejemplo más sencillo de React router</h1>;
}

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById("root")
);
```

- El ejemplo más sencillo de Router con react-router, aunque sin rutas por el momento

Switches y rutas en React router

```
<Switch>
    <Route path="/pagina1">
        <Pagina-1 />
    </Route>
    <Route path="/pagina2/:id">
        <Detalle />
    </Route>
    <Route path="/pagina2">
        <Pagina-2 />
    </Route>
    <Route path="/">
        <Inicio />
    </Route>
</Switch>
```

- El switch es el núcleo del enrutamiento
- Las rutas coinciden con las rutas de la **URL**
- Los parámetros de ruta se marcan con **dos puntos** (Ej -> **:id**)
- Hay un estricto orden de prioridad para las coincidencias

Switches y rutas en React router

```
<Switch>
    <Route path="/pagina1">
        <Pagina1 />
    </Route>
    <Route path="/pagina2/:id">
        <Detalle />
    </Route>
    <Route path="/pagina2">
        <Pagina2 />
    </Route>
    <Route path="/">
        <Inicio />
    </Route>
</Switch>
```

- El switch es el núcleo del enrutamiento
- Las rutas coinciden con las rutas de la URL
- Los parámetros de ruta se marcan con dos puntos (Ej -> :id)
- Hay un estricto orden de prioridad para las coincidencias

Switches y rutas en React router

```
<Switch>
    <Route exact path="/">
        <Inicio/>
    </Route>
    <Route path="/pagina1">
        <Pagina1/>
    </Route>
    <Route path="/pagina2/:id">
        <Detalle/>
    </Route>
    <Route path="/pagina2">
        <Pagina2/>
    </Route>
    <Route path="*">
        <NoEncontrado/>
    </Route>
</Switch>
```

- Usando la propiedad **exact** le indicamos un match estricto
- Igualmente usando el **wildcard** * en la propiedad path le indicamos que haga match con todo

Switches y rutas en React router

```
function DemoRouter() {
  return (
    <Switch>
        <Route path="/pagina1">
          <Pagina1/>
        </Route>
        <Route path="/pagina2">
          <Pagina2/>
        </Route>
        <Route path="/">
          <Inicio/>
        </Route>
    </Switch>
  )
}
```

```
function Pagina2() {
    let { path, url } = useRouteMatch();
    return(
        <div>
            <Switch>
                <Route exact path={path}>
                <h3>Seleccione un detalle.</h3>
                </Route>
                <Route path={`${path}/:detailId`}>
                <Detalle />
                </Route>
            </Switch>
        </div>
    )
}
```

- Es posible utilizar el componente **Switch** para anidar rutas como necesitemos

Navegación básica por la aplicación

```
<Link to="/pagina1#un-hash?param=valor" />

<Link
  to={location => `${location.pathname}#un-hash?param=valor`}
 />
```

```
<Link
  to={{
    pathname: "/pagina1",
    search: "?param=valor",
    hash: "#un-hash",
    state: { propiedadDeEstado: true }
  }}
/>
```

- El componente de navegación que usamos es **Link**
- La propiedad **to** indica el destino de nagevación
- Esta propiedad puede ser **un string**, **un objeto** o **una función**

Navegación básica por la aplicación

```
<Link
  to={{
    pathname: "/pagina1",
    search: "?param=valor",
    hash: "#un-hash",
    state: { propiedadDeEstado: true }
  }}
/>
```

- La propiedad **pathname** representa la localización
- **search** inyecta parámetros de búsqueda en la URL
- **hash** actúa como search pero con un hash
- **state** inyecta un valor de estado en el componente de destino

Navegación básica por la aplicación

```
<NavLink to="/faq" activeClassName="selected">
  FAQs
</NavLink>

<NavLink
  to="/pagina1/detalle2"
  isActive={(match, location) => {
    if (!match) {
      return false;
    }
    // devolvemos "true" sólo si es "detalle3"
    const id = parseInt(match.params.detailId);
    return id === 'detalle3';
  }}
>
  Event 123
</NavLink>
```

- El componente **NavLink** es similar a **Link** pero con lógica extra para cuando hace match
- La propiedad **activeClassname** determina la clase CSS
- Podemos especificar una lógica de selección más compleja con un callback en **isActive**

PARA RESUMIR

- Llamamos **Router** en react a un componente que muestra u oculta condicionalmente nuestras vistas, administrando la URL y sin realizar una recarga

- La librería de **react-router** es a día de hoy la más usada y gestiona los problemas habituales del routing de manera liviana

- El routing de forma general, y específicamente para esta librería, se divide en **switchs** y **rutas**