React intermedio

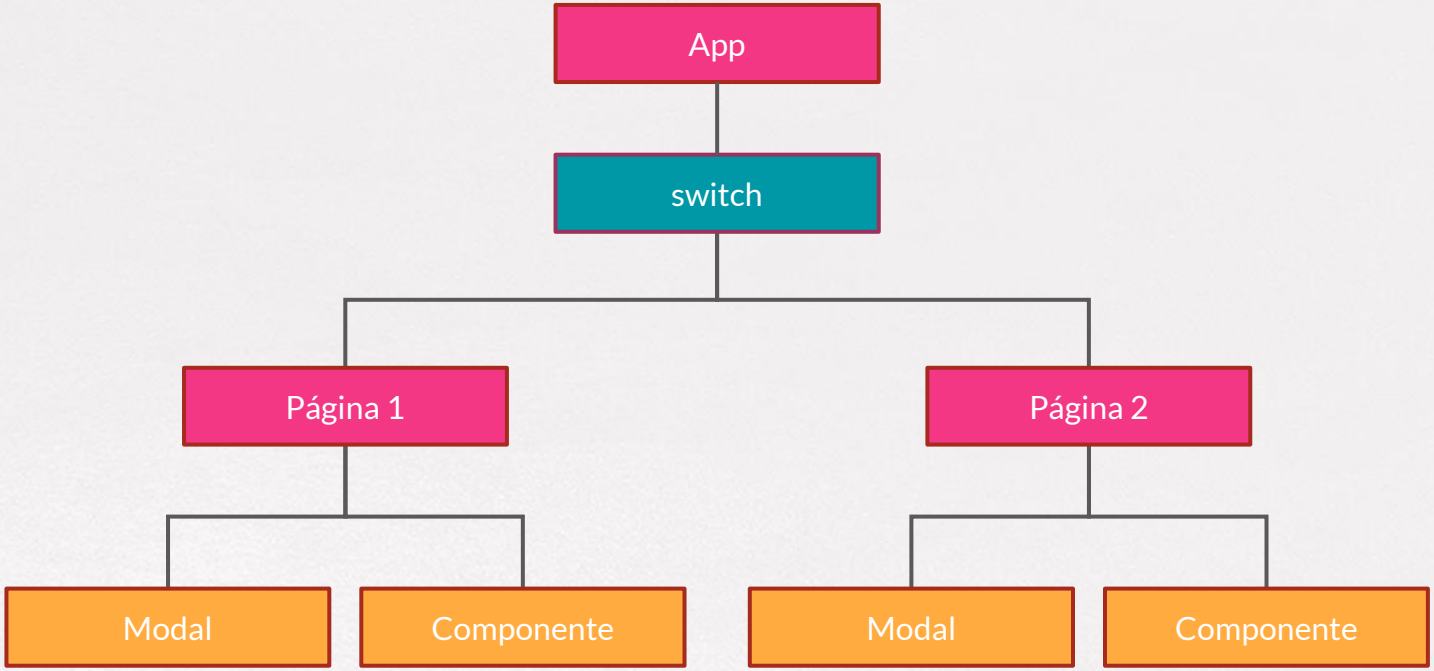
Rutas, navegación y portales

Portales en React

ÍNDICE

- El problema de la jerarquía
- El portal como solución
- Propagación de eventos en portales

El problema de la jerarquía

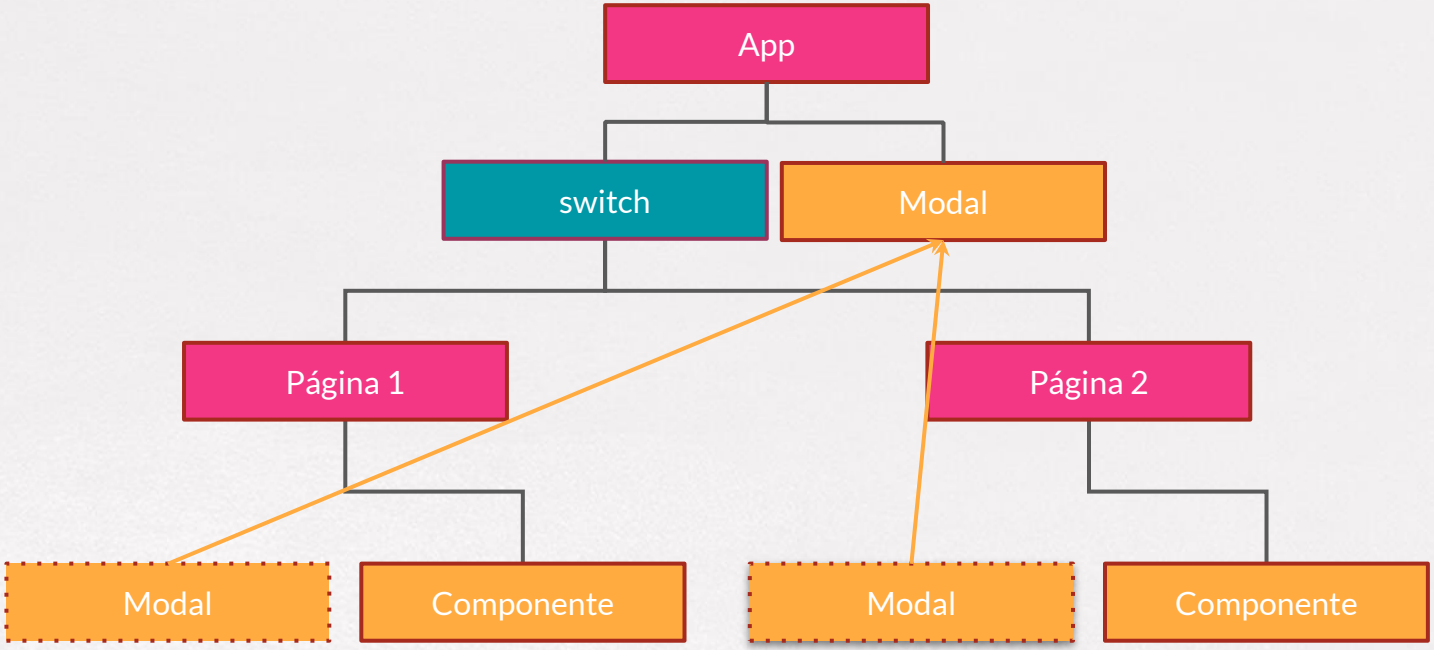


El problema de la jerarquía

```
<body>
<div id="app">
  <div id="pagina-1">
    <div class="un-componente">
        <!--El contenido del componente-->
    </div>
    <div class="modal">
        <!--El contenido de nuestro modal-->
    </div>
  </div>
</div>
</body>
```

- **La jerarquía de componentes** es correcta pero **la jerarquía en el DOM** no lo es: **Un modal siempre debería de estar en primer nivel**

El problema de la jerarquía



El problema de la jerarquía

```
<body>
<div id="app">
  <div id="modal-container">
    <div class="modal">
        <!--El contenido de nuestro modal-->
    </div>
  </div>
  <div id="pagina-1">
    <div class="un-componente">
        <!--El contenido del componente-->
    </div>
  </div>
</div>
</body>
```

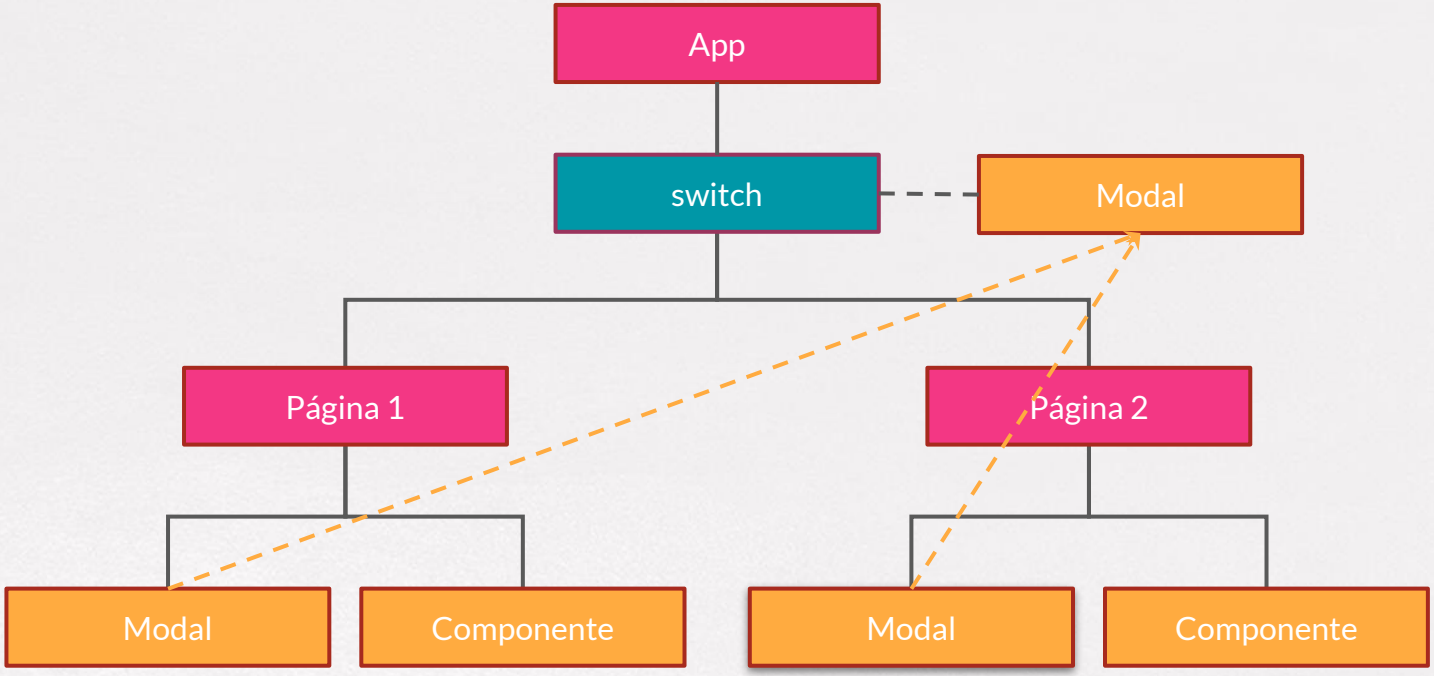
- De esta manera el DOM tiene una jerarquía correcta y el modal estaría **por encima de las vistas**

El problema de la jerarquía

```
class Pagina1 extends React.Component {
    constructor(props) {
      super(props);
      this.state = {
        modalMessage: 'Mensaje del modal'
      }
    }
    render() {
      return (
        <div className="pagina-1">
          <Componente/>
          <!-- Como no tenemos el modal disponible aquí, no podemos pasarle el mensaje del
                estado del componente como propiedad-->
        </div>
      )
    }
}
```

- Sin embargo la jerarquía para el acceso al estado de la página y la cascada de props **resulta incorrecto**

El portal como solución



El portal como solución

```
const Modal = props => {
    return ReactDOM.createPortal(
        <div className="modal">
            <div className="modal-header">
              { props.title }
            </div>
            <div className="modal-body">
              { props.content }
            </div>
        </div>,
        document.querySelector('#modal')
    );
};
```

- Un portal permite mantener la **jerarquía DOM y de componentes**
- Es útil en casos donde el z-index y otras propiedades **entorpecen la renderización**
- La función **createPortal** recibe cualquier **nodo React** y cualquier **nodo del DOM**

El portal como solución

```
class App extends React.Component {
    constructor(props) {
      super(props);
    }
    render() {
      return (
        <div className="app">
          <div id="modal"></div>
          <Switch>
            <Route path="/pagina1">
              <Pagina1/>
            </Route>
            <Route path="/pagina2">
              <Pagina2/>
            </Route>
          </Switch>
        </div>
      )
    }
}
```

```
const Modal = props => {
    return ReactDOM.createPortal(
        <div className="modal">
            <div className="modal-header">
              { props.title }
            </div>
            <div className="modal-body">
              { props.content }
            </div>
        </div>,
        document.querySelector('#modal')
    );
};
```

Propagación de eventos en portales

- Un portal puede estar **en cualquier parte** del árbol DOM pero **se comporta como un hijo de React normal**
- Un evento activado desde dentro de un portal **se propagará a los ancestros en el árbol de React**
- Por tanto, la lógica **no depende** de que el componente sea portal o no
- Esto nos permite usar la **propagación de eventos** por defecto de React

Propagación de eventos en portales

```
const Hijo = props => {
  return (
    <div className="modal">
      <button>Haz click</button>
    </div>
  );
}
```

```
const Modal = props => {
  return ReactDOM.createPortal(
      props.children,
      document.querySelector('#modal')
  );
};
```

```
<body>
  <div id="modal"></div>
  <div class="app"></div>
</body>
```

- El modal simplemente renderiza el componente dado y **lo inyecta en un nodo DOM hermano** del de la aplicación

Propagación de eventos en portales

```
class Padre extends React.Component {
  constructor(props) {
    super(props);
    this.state = {clicks: 0};
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(state => ({
      clicks: state.clicks + 1
    }));
  }

  render() {
    return (
      <div onClick={this.handleClick}>
        <Modal>
          <Hijo />
        </Modal>
      </div>
    )
  }
}
```

- El método handleClick afecta al estado del Padre
- Al no implementar un **onClick**, el evento del hijo se propagará a través del modal al padre
- De esta forma se cumple la propagación de eventos independientemente del punto de renderización

PARA RESUMIR

- Llamamos **Portal** a un componente de React que nos permite respetar la jerarquía de componentes mientras nos proporciona libertad para renderizar en cualquier punto del DOM
- Los portales son usados en casos como modales, diálogos, tooltips y demás elementos entorpecidos por el z-index o **sensibles a la posición en el DOM**
- Pese a encontrarse en puntos del DOM diferentes, **se respetan las consecuencias de la jerarquía de componentes de React**, incluyendo en la propagación de eventos