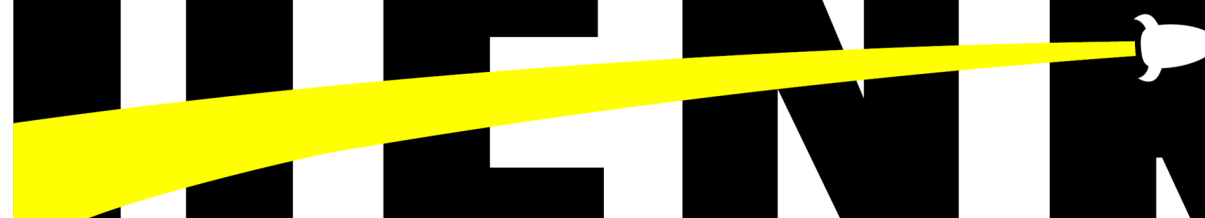


HENRY

A bright yellow beam of light originates from the left edge of the frame and tapers as it points towards the letter 'R' in the word 'HENRY'. The beam is solid yellow and has a slight gradient, appearing brighter at the tip. The word 'HENRY' is rendered in a bold, black, sans-serif font.

Bienvenidos

React – Routing

Recapitulemos...

1

Estados -> *Instancia de React Component Class, es un objeto que contiene la data y controla cómo se comporta el componente.*

2

State -> *Son las variables de estado que contienen la información.*

3

setState -> *Si queremos cambiar nuestro estado, debemos usar este método quien se encarga de actualizar la información que contiene el estado.*

4

Ciclo de vida del componente -> *Pasa por las etapas de montaje, actualización y desmontaje de los componentes.*



Recapitulemos...

5

One way data flow -> *Los datos fluyen de componentes padre a componentes hijos mediante props y componentes hijos envían eventos a componentes padre.*

6

Componentes presentacionales -> *No tienen estado. Sólo aceptan props. No contienen lógica.*

7

Componentes contenedores -> *Contienen estados para ellos o sus componentes hijos, tienen lógica*

8

Hooks -> *Nos permiten usar características de React en componentes funcionales.*



Recapitulemos...

9

useState -> *Es un hook que nos permite tener variables de estado en componentes funcionales.*

10

useEffect -> *Es un hook el cual ejecuta los ciclos de vida del componente en componentes funcionales.*





OBJETIVO DE LA CLASE

Conoceremos sobre la librería React Router donde aprenderemos cómo enrutar nuestros componentes y cómo navegar en ellos en nuestras aplicaciones en React.



¿Qué veremos hoy?

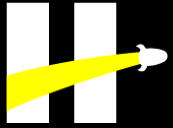
- ✓ SPA
- ✓ Routing
- ✓ Configuración
- ✓ Navegación
- ✓ Hooks: `useLocation` - `useNavigate` - `useParams`



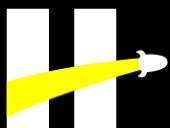


Espacio de interacción

Recuerden hacer las preguntas en el **Q&A** con contexto para que no se pierdan en el chat.

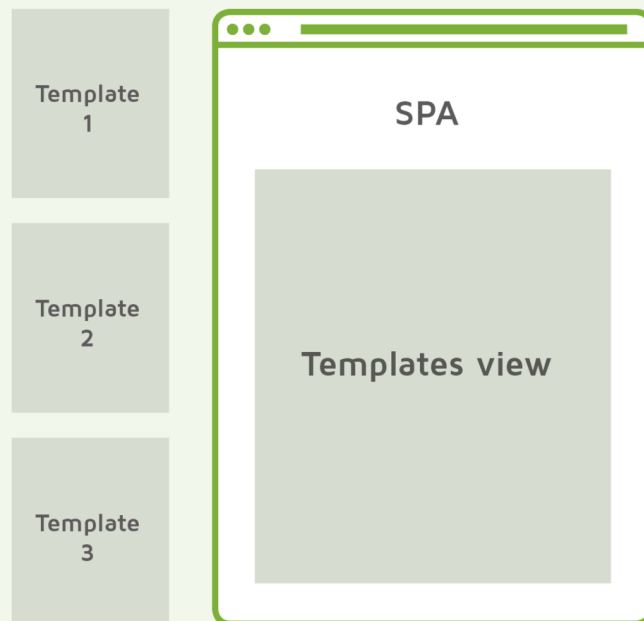


ROUTING



SPA

Single Page Application

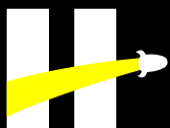


No page refresh on request

Traditional Web Application

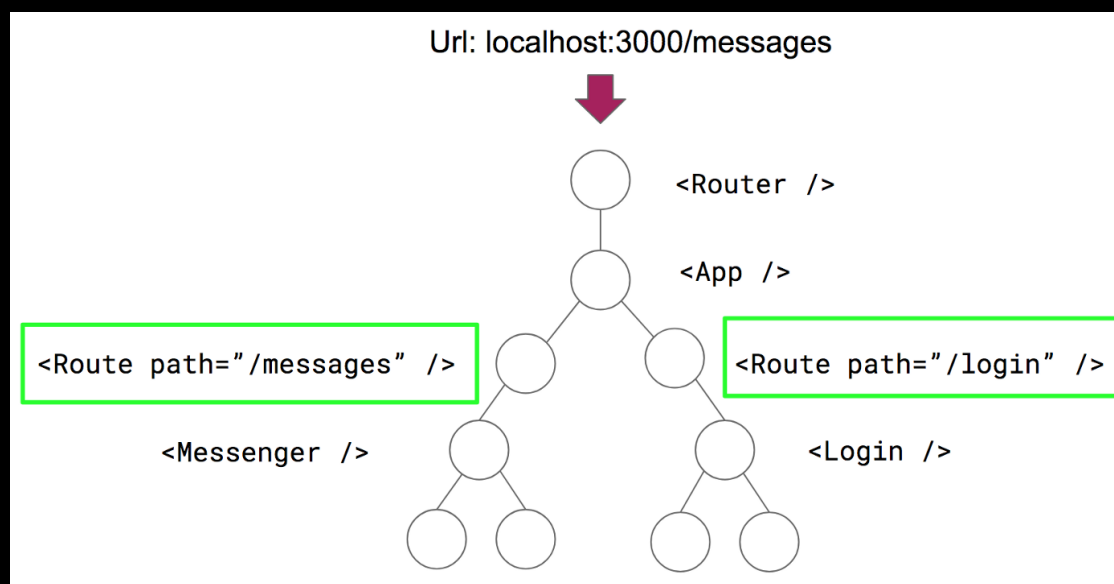


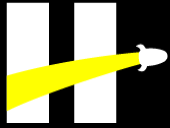
Whole page refresh on request



SPA

React Router es la librería que utilizaremos para definir de forma declarativa la vistas que queremos renderizar dependiendo la URL.

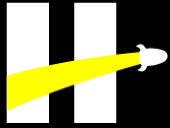




Routing

Para integrar React Router a nuestra aplicación, lo instalamos con un manejador de paquetes y luego lo importamos.

```
1 // Instalar React-Router-DOM
2 npm i react-router-dom
3
4 // Utilización
5 import { HashRouter, Route } from 'react-router-dom';
6
7 ReactDOM.render((
8   <HashRouter>
9     <App />
10  </HashRouter>
11 ), document.getElementById('app'))
```



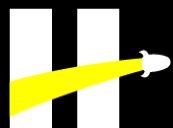
Routing

Definir las rutas de una aplicación, es un proceso que consta de dos etapas:

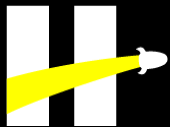
1. **Configuración:** en esta etapa definimos los nombres de las rutas, y determinamos qué componente se mostrará en cada una.
2. **Navegación:** una vez configuradas las rutas, agregamos a nuestros componentes elementos que permiten navegar de una ruta a otra.



¿Preguntas?



Configuración



Configuración

Para establecer la configuración de las rutas, es necesario importar y utilizar tres componentes principales:

- `< BrowserRouter />` envuelve a toda la aplicación y entabla la comunicación con el servidor.
- `< Routes />` agrupa todas las rutas y, ante un evento, determina cuál corresponde seguir.
- `< Route />` representa una ruta y el componente que en ella se renderiza, mediante los atributos *path* y *element*.



Configuración

```
1 import React from 'react';
2 import { createRoot } from 'react-dom/client';
3 import { BrowserRouter, Routes, Route } from 'react-router-dom';
4
5 const root = React.createRoot(document.getElementById("root"));
6
7 root.render(
8   <BrowserRouter>
9     <Routes>
10       <Route path="home" element={<Home/>} />
11       <Route path="about" element={<About/>} />
12     </Routes>
13   </BrowserRouter>
14 );
15
```



Configuración

Al definir el path de una ruta, podemos incluir parámetros variables agregando el caracter `:` a un segmento.

```
1  ...
2
3  <BrowserRouter>
4    <Routes>
5      <Route path="home" element={<Home />} />
6      <Route path="about" element={<About />} />
7      <Route path="invoices/:invoiceId" element={<Invoice />} />
8    </Routes>
9  </BrowserRouter>
10
11  ...
```



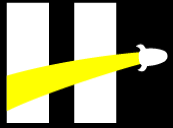
Configuración

La versión 6 de React Router permite **anidar rutas** para mantener el código ordenado y legible.

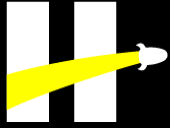
```
1 ...
2
3 <BrowserRouter>
4   <Routes>
5     <Route path="home" element={<Home />} />
6     <Route path="about" element={<About />} />
7     <Route path="sales" element={<Sales />}>
8       <Route path="invoices" element={<Invoices />}>
9         <Route path=":invoiceId" element={<Invoice />} />
10      </Route>
11      <Route path="deposits" element={<Deposits />} />
12    </Route>
13  </Routes>
14 </BrowserRouter>
15
16 ...
```



¿Preguntas?



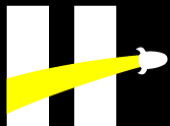
Navegación



Navegación

Configuradas las rutas, el siguiente paso es incorporar componentes que permitan navegar por toda la página:

- `< Link />` equivale a un elemento `<a>` de HTML.
- `< NavLink />` es similar al anterior, con la diferencia de que permite modificar los estilos del link cuando está activo.

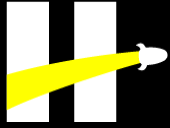


Navegación

El componente `<Link />` permite redireccionar al usuario hacia una nueva URL, que definimos mediante el atributo `to`.

```
1 import React from 'react';
2 import { Link } from 'react-router-dom';
3
4 export default function NavBar() {
5   return (
6     <div>
7       <Link to="/">Home</Link>
8       <Link to="/about">About</Link>
9       <Link to="/ejemplo">Ejemplo</Link>
10    </div>
11  );
12 };
```





Navegación

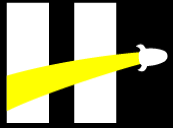
`<NavLink />` mantiene la misma funcionalidad, pero permite dar estilos al link activo recibiendo callbacks como valores de los atributos `className` y `style`.

```
1 export default function NavBar() {
2   return (
3     <div>
4       <NavLink to="/">Home</NavLink>
5       <NavLink
6         to="/about"
7         className={({ isActive }) => (isActive ? "active" : undefined)}
8       >About</NavLink>
9       <NavLink
10        to="/ejemplo"
11        style={({ isActive }) => isActive ? { color: "blue" } : null }
12      >Ejemplo</NavLink>
13    </div>
14  );
15 };
```

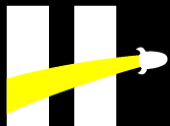




¿Preguntas?



Hooks

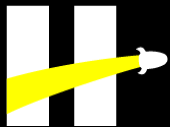


useLocation

```
1 import React from 'react';
2 import { useLocation } from 'react-router-dom';
3
4 export default function Location() {
5   let location = useLocation();
6
7   ...
8 };
```

```
Location:
▼ Object { pathname: "/location", state: {...}, search: "",
hash: "" }
  hash: ""
  pathname: "/location"
  search: ""
  ▼ state: Object { extraData: "Henry" }
    extraData: "Henry"
```

Retorna el objeto **Location**, que contiene información sobre nuestra posición dentro de las rutas de la aplicación.

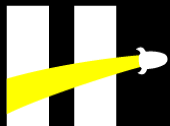


useNavigate

```
1 import { useNavigate } from "react-router-dom";
2
3 export default function Navigation() {
4   const navigate = useNavigate();
5
6   useEffect(() => {
7     if (userIsInactive) {
8       navigate("/session-timed-out");
9     }
10  }, [userIsInactive]);
11
12  ...
13 };
```

```
▼ function navigate(to, options) ↗
  length: 2
  name: ""
  ▶ prototype: Object { ... }
  ▶ <prototype>: function ()
```


Retorna una función **navigate** que permite navegar por el historial de navegación de la ventana.



useParams

```
1 import React from 'react';
2 import { useParams } from 'react-router-dom';
3
4 export default function Mostrar() {
5   let params = useParams();
6   return (<span>Estoy en la ciudad {params.ciudadId}</span>)
7 };

```

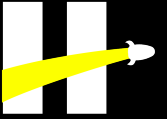
Params:  Object { ciudadId: "5" }
| ciudadId: "5"
| ▶ <prototype>: Object { ... }

Cuando la URL especificada dentro de <Route> tiene segmentos dinámicos (**params**), useParams permite obtener el o los valores pasados como parámetro dentro del path.



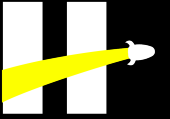
¿Preguntas?

Resumen



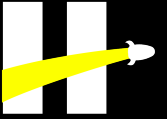
- **SPA:** Se recarga una vez, a partir de allí navega entre los componentes.
- **Routing:** Aprendimos que el enrutamiento es un proceso de enlaces de una dirección URL a un recurso específico de la aplicación, enlazando una URL a un componente de React. También aprendimos que para aplicar este enrutamiento es necesario instalar, configurar y utilizar la librería react-router-dom.
- **BrowserRouter:** Componente que envuelve a toda la aplicación.
- **Routes:** Genera un árbol de rutas y permite que renderizar un componente que coincida con la URL de la barra de navegación.

Resumen



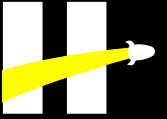
- **Route:** Representa una ruta en el árbol, necesita de las propiedades path y element para representar una ruta.
- **Rutas anidadas:** Suceden cuando un componente Route es contenedor o padre de otro componente Route.
- **Rutas dinámicas:** Significa que este puede coincidir con cualquier valor en dicho segmento, se representan con dos punticos antes del nombre de la ruta.
- **Navegación:** Para que los usuarios puedan navegar en nuestra aplicación, precisamos de los componentes Link o NavLink.

Resumen



- **Link:** Este componente es similar al elemento anchor o etiqueta `<a>` `` en HTML y nos sirve para linkear o enrutar dentro de componentes. Tiene la propiedad `“to”`.
- **NavLink:** Hace lo mismo que Link, solo que podemos aplicar estilos que señalen la ruta activa en la que se encuentra el usuario.
- **useLocation:** Es un hook de React Router, el cual es una función que nos devuelve un objeto con la información sobre la dirección URL activa o actual, cada vez que el usuario cambie la dirección URL, nos devuelve un nuevo objeto location.

Resumen



- **useNavigate:** Es un hook que retorna una función que nos permite navegar de forma programática dentro de la aplicación, navigate recibe un parámetro en el que puede ser una ruta específica o puede ser un valor delta (número) en el que equivale navegar hacia atrás si es negativo o adelante si es positivo.
- **useParams:** Es un hook que nos permite acceder a parámetros dinámicos en la URL. Nos devuelve un objeto con el valor del parámetro de la URL.



Homework



01- Exercises

Implementar rutas, redirecciones a una aplicación ya hecha



02- Integration

Implementar enrutamiento a nuestra aplicación Rick & Morty



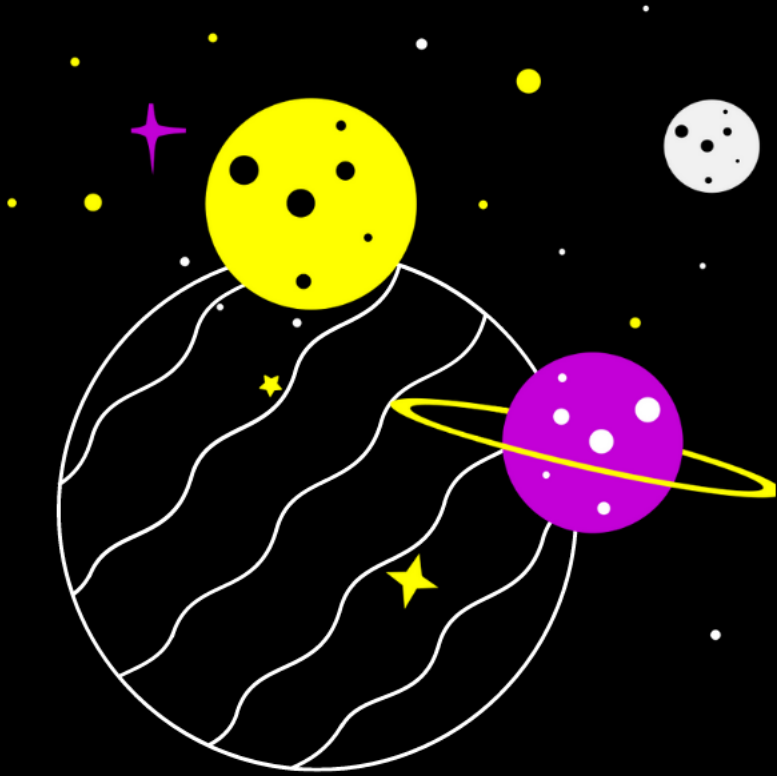
¿Preguntas?



PROXIMA CLASE

React - Forms

HENRY



¡Muchas gracias!