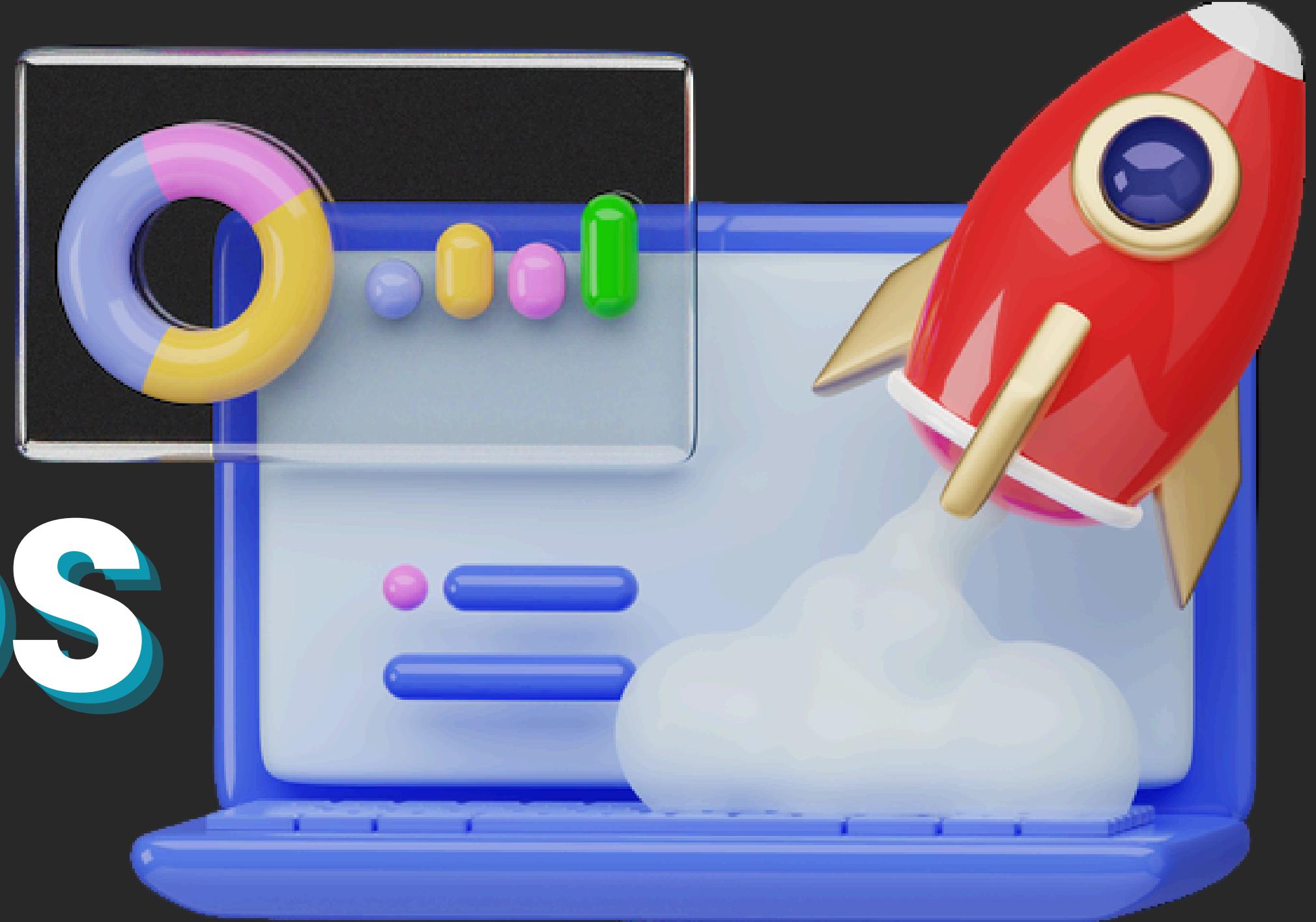
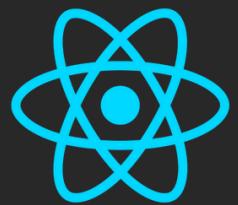


ESTADOS Y RUTAS EN REACT

Por:

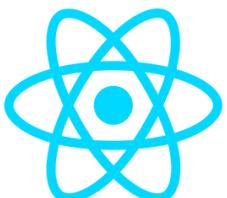
Keyla Dayana Arboleda Mina
Carlos Andres Cifuentes Montaño
Jose Fernando Sinisterra Ibargüen
Darío Restrepo Landázury

LOS ESTADOS



¿QUÉ SON?

En **React**, los componentes a menudo necesitan cambiar lo que se muestra en pantalla como resultado de una interacción. Para manejar el estado en **React** se utiliza el hook **useState** que permite agregar variables de estados a los componentes.

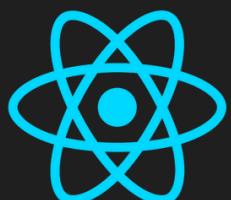


ESTADO LOCAL

Este se maneja usando el **hook *useState*** y es específico para cada componente. Un ejemplo claro es el componente **Avatar** que maneja un estado local para controlar si la descripción está expandida. A continuación, la implementación de lo mencionado:



```
1 const [isExpanded, setIsExpanded] = useState(false);
2
3 const toggleExpand = () => {
4     setIsExpanded(!isExpanded);
5 }
```



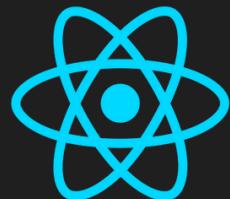
ESTADO GLOBAL

Es un estado compartido accesible por múltiples componentes, no solo padre-hijo profundo. Para su acceso, se puede utilizar **Redux**.



```
1 export const ParticipantsSlice = createSlice({
2   name: 'participants',
3   initialState,
4   reducers: {
5     incrementLike: (state,action) => {
6       const participant = state.participants.find(participant => participant.id === action.payload);
7       if (participant) {
8         participant.like ++
9       }
10      }
11    }
12  });
```

El ejemplo anterior implementa el manejo del estado global de un listado de participantes con su conteo de likes.



FLUJO DE DATOS

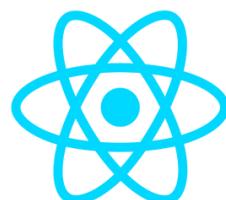
En **React**, se da de manera unidireccional donde los datos fluyen siempre desde los componentes padre hacia los hijos mediante props. Esto trae consigo algunos beneficios.

Predecibilidad: hace los cambios de estado más previsibles; las acciones explícitas y reducers definen las transiciones.

Mantenibilidad: separar estado y lógica (reducers) produce código más limpio y menos errores.

Escalabilidad: controla actualizaciones de estado en aplicaciones grandes, evitando problemas del enlace bidireccional.

Depuración: facilita rastrear cambios de estado; **Redux DevTools** permite ver acciones, estados en el tiempo y reproducir errores.

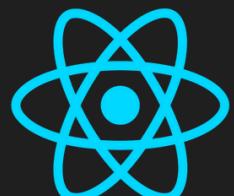


GESTIÓN DEL ESTADO

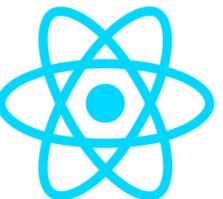


Para el acceso y modificación del estado global, se usan los hooks **useSelector** y **useDispatch**. Por ejemplo, el código muestra una implementación para gestionar el estado de los likes del componente **Button**.

```
1  export default function Button({ participantId, children, disabled = false }) {  
2    const dispatch = useDispatch();  
3    const participant = useSelector(state =>  
4      state.participants.participants.find(p => p.id === participantId)  
5    );  
6    const count = participant?.like || 0;  
7  
8    const getButtonColor = () => {  
9      if (count <= 3) return '#0a4188ff';  
10     if (count <= 7) return '#af9609ff';  
11     return '#0cb991ff';  
12   };  
13  
14   function like() {  
15     if (count < 10) {  
16       dispatch(incrementLike(participantId));  
17     }  
18   }  
19 }
```



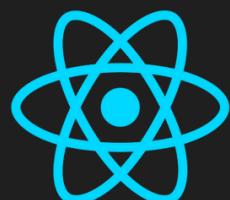
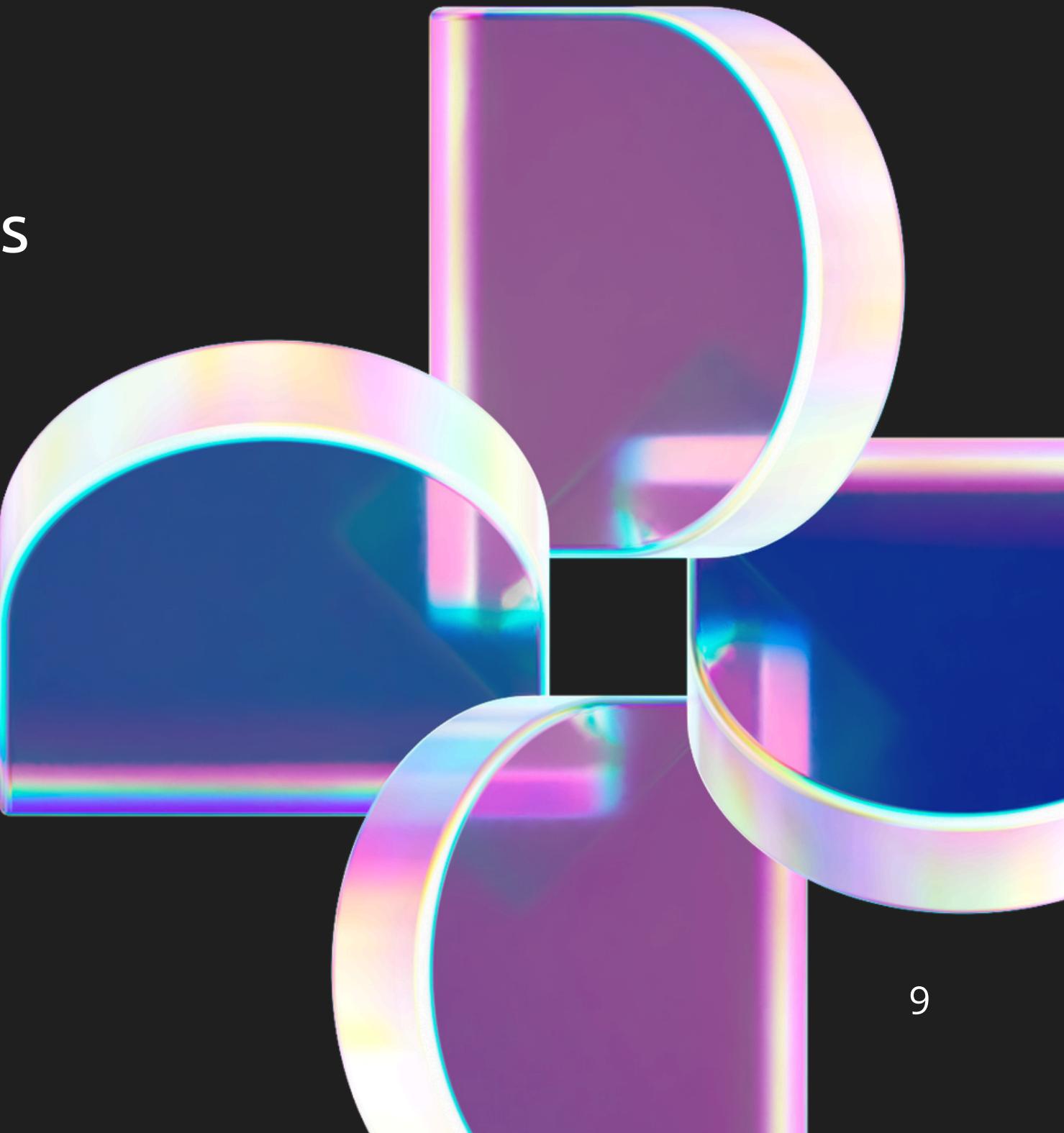
RUTAS Y NAVEGACIÓN



¿QUÉ ES EL ENRUTAMIENTO?

Este es el proceso de definir cómo una aplicación web maneja diferentes URL's o rutas que el usuario puede visitar.

En **React**, se utiliza la herramienta **React Router DOM** para la creación de una navegación fluida, mejorando tanto la experiencia del usuario como el rendimiento general de la aplicación web.



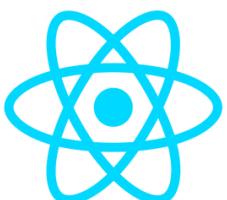
INSTALACIÓN

Para instalar y utilizar **React Router-DOM** debemos ejecutar el siguiente comando:

npm install react-router-dom

Después de la instalación, debemos verificar en el **package.json** de nuestro proyecto que esté la dependencia allí, sería algo así:

```
1 "dependencies": {  
2   "@reduxjs/toolkit": "^2.9.0",  
3   "react": "^19.1.1",  
4   "react-dom": "^19.1.1",  
5   "react-redux": "^9.2.0",  
6   "react-router-dom": "^6.30.1"  
7 }
```



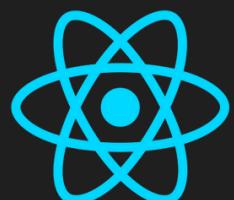
APLICACIÓN Y USO

Lo primero que se tiene que hacer es importar **React Router-DOM** en nuestro componente para así poder definir rutas. Con la siguiente línea se logra lo anterior:



```
1 import { BrowserRouter, Routes, Route, Link } from 'react-router-dom'
```

- Con **BrowserRouter** habilitamos el enrutamiento del lado del cliente.
- Con **Link**, definimos la relación entre el documento actual y los demás componentes.
- **Route** establece una ruta específica y la asocia con una que coincida.
- **Routes** actúa como contenedor para agrupar rutas individuales.



EJEMPLO DE USO

En el **nav** contenemos las rutas que podemos navegar nuestra página.



```
1 <nav>
2   <h3><Link to="/" >Inicio</Link></h3>
3   <h3><Link to="/nosotros" >Nosotros</Link></h3>
4   <h3><Link to="/ruta" >Links</Link></h3>
5 </nav>
```

Aquí se contienen las rutas, tanto **privadas** como **públicas**:



```
1 <Routes>
2   <Route path="/login" element={<Login />} />
3   <Route path="/" element={<div className='MensajeInicio'>
4     <h2>Explora nuestras secciones para ver más contenido.</h2>
5   </div>} />
6   <Route path="/nosotros" element={
7     <PrivateRoute>
8       <Nosotros />
9     </PrivateRoute>
10   } />
11   <Route path="/ruta" element={
12     <PrivateRoute>
13       <Ruta />
14     </PrivateRoute>
15   } />
16 </Routes>
```

MUCHAS GRACIAS

