Estados Globales en React

Centro de Servicios y Gestión Empresarial
SENA Regional Antioquia

@SENAComunica

www.sena.edu.co

Concepto – Estados globales

El estado global es la información que debe ser compartida entre varios componentes de una aplicación, a diferencia del estado local (useState), que pertenece a un solo componente.

Este tipo de estado permite que múltiples partes de la interfaz accedan o actualicen los mismos datos, lo cual es útil en situaciones como:

- Usuario autenticado
- Productos del carrito
- Preferencias de idioma o tema
- Notificaciones.

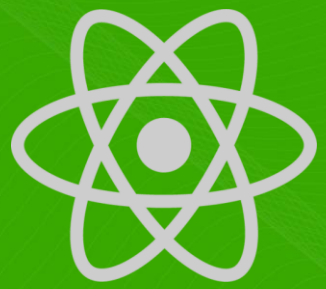
Concepto – Estados globales

En React, los estados globales se gestionan mediante soluciones nativas como el Context API o mediante bibliotecas externas de administración de estado como Redux, Zustand o Recoil.

Estas herramientas permiten manejar datos comunes en diferentes niveles de la jerarquía de la aplicación de forma eficiente y estructurada.

Concepto – Estados globales

Alternativa	Uso principal	Limitaciones
useState	Estado local simple	No es compartido entre componentes
useReducer	Estado local complejo	No es global, requiere Context API
Redux Toolkit	Estado global, compartido y escalable	Requiere configuración inicial



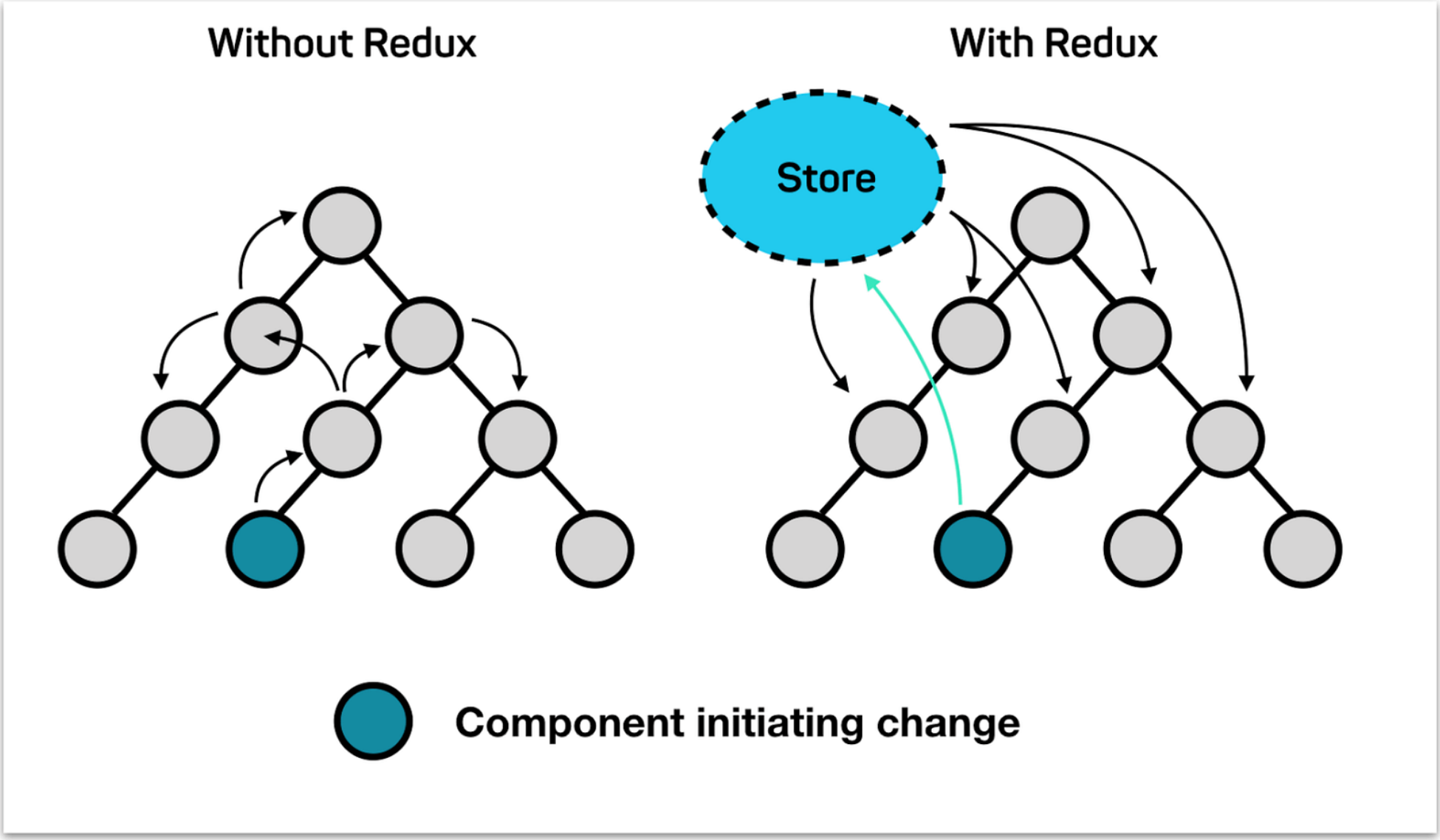
Biblioteca de gestión de estados

Bibliotecas de gestión de estados

Redux es una herramienta para gestionar el estado global de una aplicación utilizando una arquitectura predecible basada en el patrón **acción-reductor-estado**:

- **Acciones**: representan eventos que describen algo que ha ocurrido en la aplicación.
- **Reductores**: son funciones que reciben el estado actual y una acción, y devuelven un nuevo estado.
- **Store**: almacena el estado de la aplicación y lo pone a disposición de los componentes.

Concepto – Estados globales



Bibliotecas de gestión de estados

Redux Toolkit es la forma moderna, oficial y recomendada de trabajar con Redux, donde simplifica considerablemente la creación y gestión de: El estado global (store), Reductores y Acciones.

Además, facilita el manejo de peticiones asíncronas y mejora las herramientas de depuración, haciendo que el desarrollo con Redux sea más eficiente y menos propenso a errores.

Redux- Instalación

Biblioteca: Instalar las dependencias necesarias.

```
npm install @reduxjs/toolkit react-redux
```

- Instala Redux Toolkit (`@reduxjs/toolkit`), que incluye redux internamente como dependencia.
- Instala React Redux (`react-redux`), que conecta Redux con componentes de React.

Redux- Estructura

```
src/
├── app/
│   └── store.js              # Configuración del store
├── features/
│   └── auth/
│       ├── authSlice.js      # Slice del estado de autenticación
│       └── Login.jsx         # Componente que lo usa
```

```
import { createSlice } from '@reduxjs/toolkit';

const nombreDelSlice = createSlice({
  name: 'nombre',                  // Nombre del slice
  initialState: {                  // Estado inicial
    propiedad1: valor1,
    propiedad2: valor2
  },
  reducers: {                      // Funciones que modifican el estado
    nombreDeLaAccion: (state, action) => {
      // lógica para modificar el estado
    },
    otraAccion: (state, action) => {
      // otra lógica de actualización
    }
  }
});

// Exportar acciones y reducer
export const { nombreDeLaAccion, otraAccion } = nombreDelSlice.actions;
export default nombreDelSlice.reducer;
```

Redux- Estructura

```
import { configureStore } from '@reduxjs/toolkit';
import nombreReducer from 'ruta/al/slice’;


export const store = configureStore({
  reducer: {
    // Asociación entre clave y reducer importado
    claveDelEstado: nombreReducer,
    // puedes agregar más slices aquí
  },
  // Opcional: configuración extra (middleware, devTools, etc.)
});
```

Comparativa

Característica	useReducer	Redux Toolkit
Parte de React	Nativo	Librería externa
Tipo de estado	Local por componente	Global para toda la app
Compartir estado	Solo con Context	Naturalmente global
Escalabilidad	Limitada	Alta (múltiples slices)
DevTools	No	Sí
Middleware / async	Manual	Integrado (createAsyncThunk)

Ejercicio de Aplicación

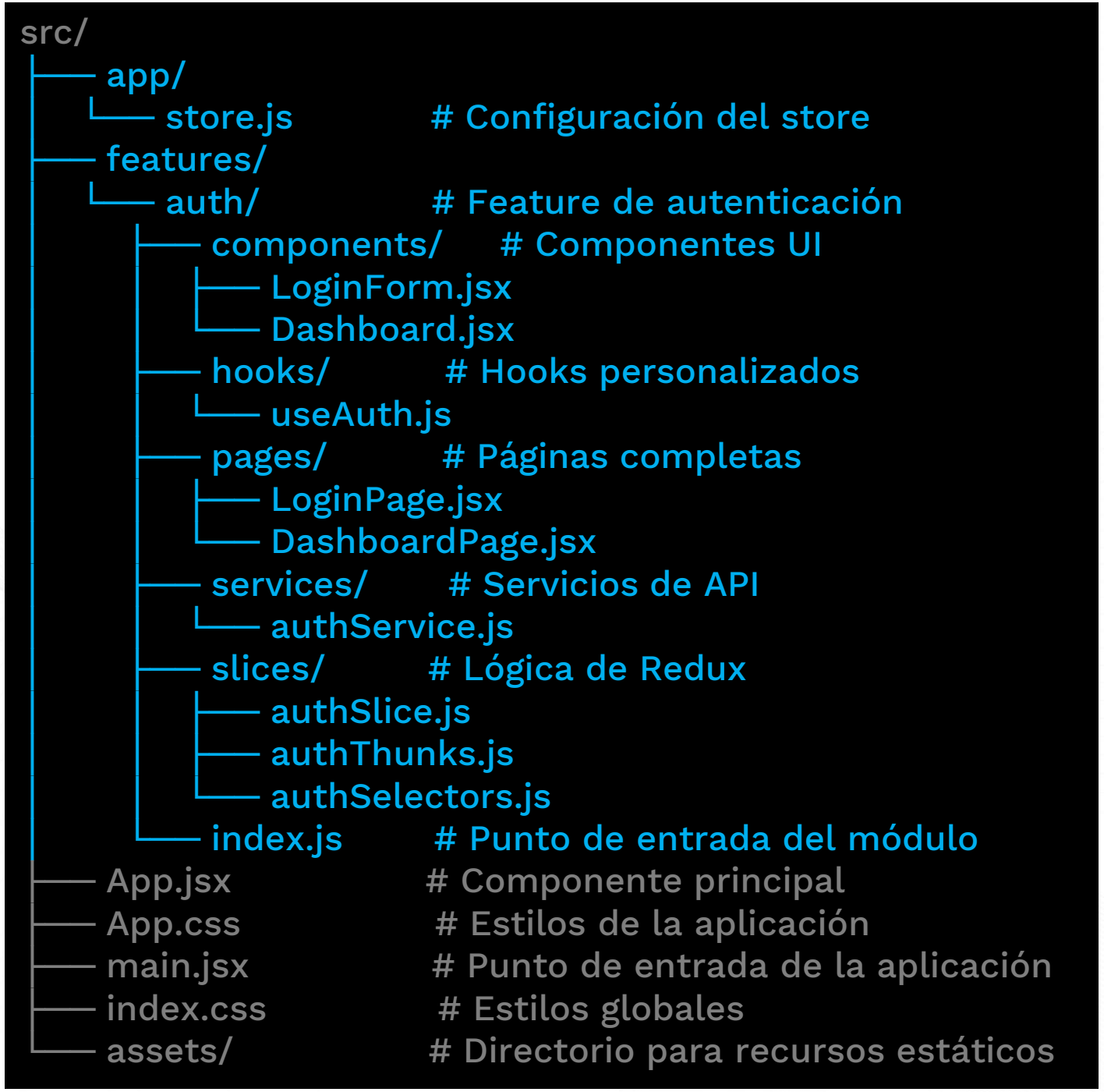
Inicio de Sesión

Descripción

Desarrollar un sistema de autenticación completo utilizando React y Redux Toolkit, siguiendo una estructura de proyecto modular y escalable. Este ejercicio permitirá practicar la gestión de estado global, llamadas a API, y organización de código en una aplicación React moderna.

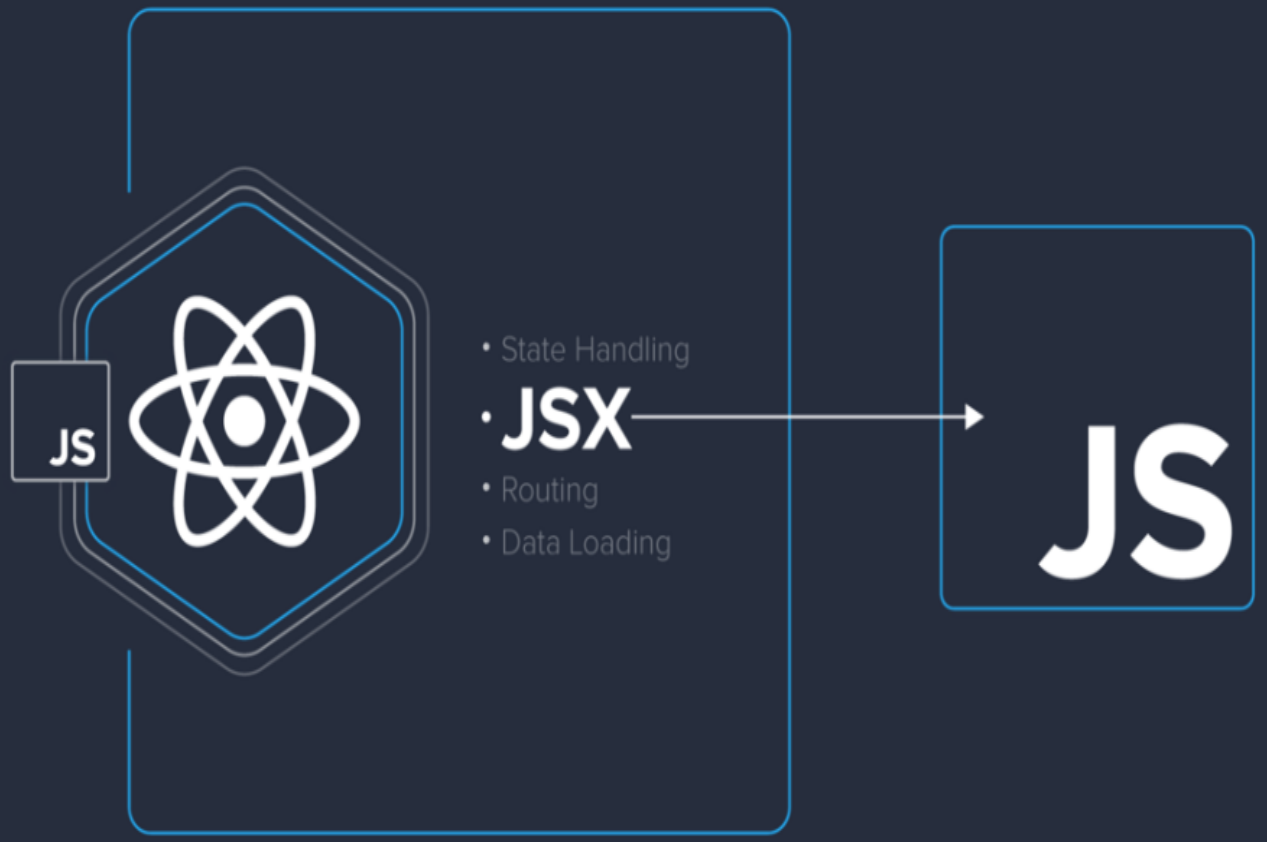
Especificaciones Técnicas

- **Tecnologías a utilizar:** React, Redux Toolkit y Fetch API para llamadas HTTP
- **API de autenticación:** https://api.escuelajs.co/api/v1/users



Paso 01: Estructura de Carpetas

Convenciones de archivos



En los proyectos de React, aunque tanto `.js` como `.jsx` son técnicamente válidos para cualquier archivo JavaScript, se suele seguir una convención para mantener el código organizado y fácil de entender.

Convenciones de archivos

.jsx: Para componentes que contienen JSX
Se usa la extensión **.jsx** para componentes de React que retornan JSX, es decir, que contienen sintaxis similar a HTML (como , <h1>, etc.).Esto ayuda a dejar claro que ese archivo contiene lógica de UI, lo que facilita su identificación durante el desarrollo.

Ejemplo

```
// Header.jsx
function Header() {
  return <h1>Bienvenido a la app</h1>;
}

export default Header;
```

Convenciones de archivos

.js: Para archivos de JavaScript puro

- Se utiliza .js para módulos que no incluyen JSX, como:
- Archivos de configuración (store.js, routes.js)
- Servicios (api.js, authService.js)
- Lógica de estado como slices de Redux Toolkit (userSlice.js)
- Funciones utilitarias (formatDate.js, validateForm.js)

Ejemplo

```
// api.js
export const getUserById = async (id) => {
  const res = await fetch(`/api/users/${id}`);
  return res.json();
};
```

Componentes

Inicio de sesión

- Desarrollar un formulario que solicite email y contraseña
- Validar campos antes de enviar
- Mostrar mensajes de error claros
- Indicador de carga durante la autenticación
- Redireccionar a Dashboard tras inicio de sesión exitoso

Gestión de Estados

Gestión del Estado de Autenticación

Implementar un slice de Redux con los siguientes estados:

- **isLoading:** Estado de carga durante la autenticación
- **isAuthenticated:** Boolean que indica si el usuario está autenticado
- **user:** Objeto con la información del usuario autenticado
- **error:** Mensaje de error si la autenticación falla

Hooks

Custom Hook

Desarrollar un hook personalizado useAuth que:

- Proporcione acceso a los estados de autenticación
- Exponga métodos para login y logout
- Facilite la reutilización de la lógica de autenticación

Servicios

Servicios de API

Implementar un servicio para autenticación que:

- Realice una llamada GET a la API para obtener usuarios
- Compare las credenciales proporcionadas con las de la API
- Retorne la información del usuario o un error

G R A C I A S

Presentó: Alvaro Pérez Niño
Instructor Técnico
Correo: aperezn@misena.edu.co
http://centrodeserviciosygestionempresarial.blogspot.com/