O que é React Redux?

React Redux é a **biblioteca oficial** que conecta o **Redux** ao **React**.
Redux é um *state manager* — um gerenciador de estado global da aplicação.

✔ Problema que ele resolve

Em aplicações React maiores, passar dados por *props* entre muitos componentes vira um caos. O Redux resolve isso ao oferecer:

- **Estado global único** (store)
- **Regras claras** para atualizar o estado (reducers)
- **Previsibilidade** — você sempre sabe como o estado muda
- **Ferramentas de debug** poderosas (Redux DevTools)

✔ Conceitos principais do Redux

Conceito	O que é
Store	Onde o estado global fica armazenado
Action	Uma intenção de mudança (ex.: `{type: "ADD_TODO"}`)
Reducer	Função pura que recebe o estado atual + action e devolve o novo estado
Dispatch	Função que dispara uma action
Selector	Função que "busca" algo dentro da store

EXEMPLO COMPLETO – React + Redux (sem Toolkit)

Vamos criar um contador com:

- Estado global
- Actions

- Reducer
- Store
- Dispatch
- Selectors
- Provider
- Componente conectado ao Redux

📂 Estrutura do projeto

```
src/
 ├─ store/
 │   ├─ actions.js
 │   ├─ reducer.js
 │   └─ store.js
 ├─ components/
 │   └─ Contador.js
 ├─ App.js
 └─ index.js
```

1 Criando as Actions

Arquivo: `src/store/actions.js`

```
// ACTION TYPES
export const INCREMENTAR = "INCREMENTAR";
export const DECREMENTAR = "DECREMENTAR";
export const ADICIONAR = "ADICIONAR";

// ACTION CREATORS
export function incrementar() {
  return { type: INCREMENTAR };
}

export function decrementar() {
  return { type: DECREMENTAR };
}
```

```
export function adicionar(valor) {
  return {
    type: ADICIONAR,
    payload: valor
  };
}
```

✔ O que está acontecendo?

- **Action type** → texto que representa a ação.
- **Action creator** → função que retorna o objeto da action.

2 Criando o Reducer

Arquivo: `src/store/reducer.js`

```
import { INCREMENTAR, DECREMENTAR, ADICIONAR } from "./actions";

const initialState = {
  valor: 0
};

// reducer SEMPRE recebe: (estadoAtual, action)
export function contadorReducer(state = initialState, action) {
  switch (action.type) {
    case INCREMENTAR:
      return { ...state, valor: state.valor + 1 };

    case DECREMENTAR:
      return { ...state, valor: state.valor - 1 };

    case ADICIONAR:
      return { ...state, valor: state.valor + action.payload };

    default:
      return state; // importante!
  }
}
```

✔ Explicação

- O reducer **não pode modificar o estado diretamente**.
- Ele deve **retornar um novo objeto**.
- O `switch` decide qual alteração fazer.
- `initialState` é usado caso o estado ainda não exista.

3 Criando a Store

Arquivo: `src/store/store.js`

```
import { createStore } from "redux";
import { contadorReducer } from "./reducer";

export const store = createStore(contadorReducer);
```

✔ O que acontece aqui?

- A store guarda o estado global.
- `createStore()` recebe o reducer principal.

4 Conectando Redux ao React

Arquivo: `src/index.js`

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./App";
import { Provider } from "react-redux";
import { store } from "./store/store";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(
```

```
  <Provider store={store}>
    <App />
  </Provider>
);
```

✔ Explicação

`Provider` torna a store acessível a **todos os componentes** React.

5 Criando o componente que usa Redux

Arquivo: `src/components/Contador.js`

```
import React from "react";
import { useDispatch, useSelector } from "react-redux";
import { incrementar, decrementar, adicionar } from
"../store/actions";

function Contador() {
  // LER valor do estado GLOBAL
  const valor = useSelector((state) => state.valor);

  // Função que dispara ACTIONS
  const dispatch = useDispatch();

  return (
    <div>
      <h2>Valor atual: {valor}</h2>

      <button onClick={() => dispatch(incrementar())}>+</button>
      <button onClick={() => dispatch(decrementar())}>-</button>

      <button onClick={() => dispatch(adicionar(5))}>
        Adicionar 5
      </button>
    </div>
  );
}

export default Contador;
```

✔ O que está acontecendo?

- `useSelector()` → lê dados da store.
- `useDispatch()` → envia ações para o reducer.
- `incrementar()` → action creator
- Redux envia a action para o reducer → novo estado é criado → React re-renderiza o componente.

6.Usando o componente no App

Arquivo: `src/App.js`

```
import Contador from "./components/Contador";

function App() {
  return (
    <div style={{ padding: 20 }}>
      <h1>Exemplo Redux</h1>
      <Contador />
    </div>
  );
}

export default App;
```

Fluxo Completo Resumido

1. **Usuário clica no botão**
2. O componente executa `dispatch({type: ...})`
3. A store envia essa action para o reducer
4. O reducer cria e retorna um novo estado