



Gerenciamento de Estados com Redux

Douglas Silva de Melo

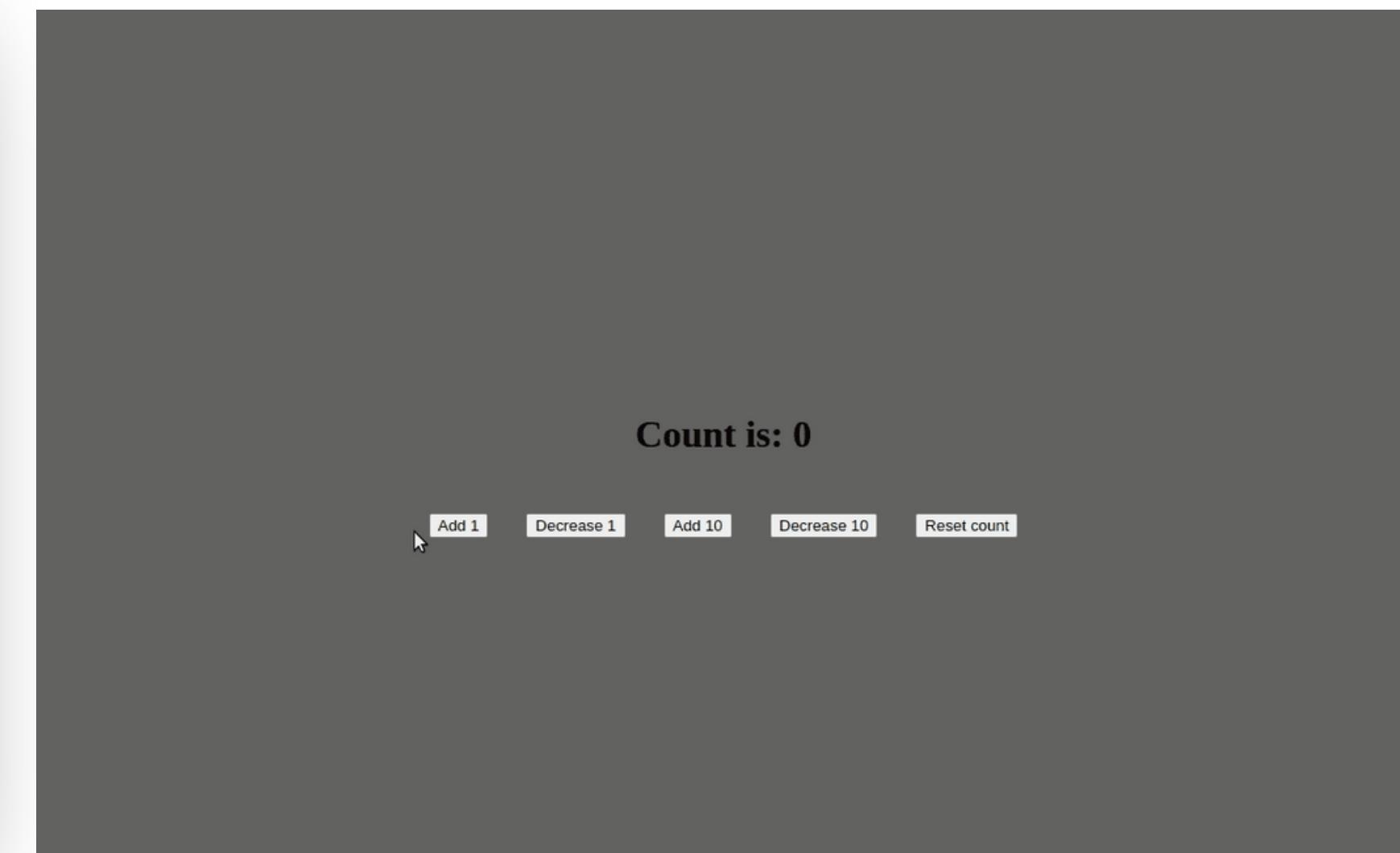


UFAM

Gerenciamento de Estados

- O estado é um objeto que contém informações sobre um determinado componente.
- Para implementar o estado em nossos componentes, o React nos fornece um gancho chamado useState.

```
● ● ●  
// App.js  
import { useState } from 'react'  
  
function App() {  
  
  const [count, setCount] = useState(0)  
  
  return (  
    <div className="App">  
      <p>Count is: {count}</p>  
  
      <div>  
        <button onClick={() => setCount(count+1)}>Add 1</button>  
        <button onClick={() => setCount(count-1)}>Decrease 1</button>  
  
        <button onClick={() => setCount(count+10)}>Add 10</button>  
        <button onClick={() => setCount(count-10)}>Decrease 10</button>  
  
        <button onClick={() => setCount(0)}>Reset count</button>  
      </div>  
    </div>  
  )  
  
  export default App
```



E quando temos um projeto maior?

- Muitos componentes compartilham o **mesmo estado** (Toda vez que muda o estado em um, tem renderizar todos).
- Todos os componentes são renderizados novamente.
- Os **redutores** podem resolver este problema.

E quando temos um projeto maior?

- Muitos componentes compartilham o **mesmo estado** (Toda vez que muda o estado em um, tem renderizar todos).
- Todos os componentes são renderizados novamente.
- Os **reduidores** podem resolver este problema.

```
// App.js
import { useReducer } from 'react'
import './App.scss'

function App() {

  function reducer(state, action) {
    switch (action.type) {
      case 'ADD': return { count: state.count + 1 }
      case 'SUB': return { count: state.count - 1 }
      case 'ADD10': return { count: state.count + 10 }
      case 'SUB10': return { count: state.count - 10 }
      case 'RESET': return { count: 0 }
      default: return state
    }
  }

  const [state, dispatch] = useReducer(reducer, { count: 0 })

  return (
    <div className="App">
      <p>Count is: {state.count}</p>

      <div>
        <button onClick={() => dispatch({type: 'ADD'})}>Add 1</button>
        <button onClick={() => dispatch({type: 'SUB'})}>Decrease 1</button>
        <button onClick={() => dispatch({type: 'ADD10'})}>Add 10</button>
        <button onClick={() => dispatch({type: 'SUB10'})}>Decrease 10</button>
        <button onClick={() => dispatch({type: 'RESET'})}>Reset count</button>
      </div>
    </div>
  )
}

export default App
```

1) npx create-react-app redux_teste --template typescript
2) cd redux_teste
2) npm i react-redux @reduxjs/toolkit

E quando temos um projeto maior?

- Muitos componentes compartilham o **mesmo estado** (Toda vez que muda o estado em um, tem que renderizar todos).
- Todos os componentes são renderizados novamente.
- Os **redutores** podem resolver este problema.

```
// App.js
import { useReducer } from 'react'
import './App.scss'

function App() {
  function reducer(state, action) {
    switch (action.type) {
      case 'ADD': return { count: state.count + 1 }
      case 'SUB': return { count: state.count - 1 }
      case 'ADD10': return { count: state.count + 10 }
      case 'SUB10': return { count: state.count - 10 }
      case 'RESET': return { count: 0 }
      default: return state
    }
  }

  const [state, dispatch] = useReducer(reducer, { count: 0 })

  return (
    <div className="App">
      <p>Count is: {state.count}</p>

      <div>
        <button onClick={() => dispatch({type: 'ADD'})}>Add 1</button>
        <button onClick={() => dispatch({type: 'SUB'})}>Decrease 1</button>
        <button onClick={() => dispatch({type: 'ADD10'})}>Add 10</button>
        <button onClick={() => dispatch({type: 'SUB10'})}>Decrease 10</button>
        <button onClick={() => dispatch({type: 'RESET'})}>Reset count</button>
      </div>
    </div>
  )
}

export default App
```

Um **redutor(reducer)**, que é a função que irá consolidar todas as mudanças de estado possíveis

Uma função de envio (*dispatch*), que enviará as ações de modificação para o redutor.

Torna o gerenciamento de estado mais modular e previsível.

E o Redux?

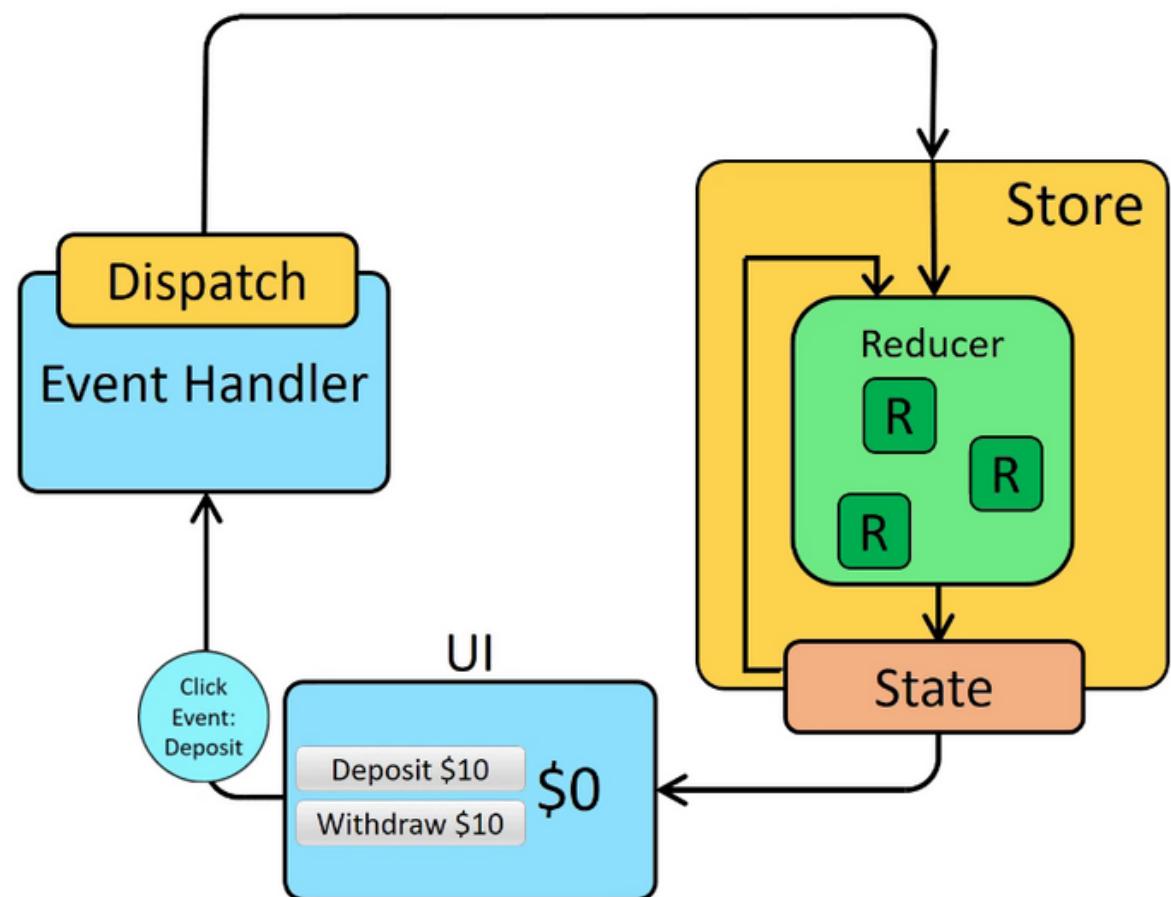
- Redux serve para gerenciar o estado de um aplicativo de forma centralizada, previsível e escalável.
- Ele oferece uma abordagem para armazenar e controlar o estado global do aplicativo.
- Uma biblioteca agnóstica, o que significa que pode ser implementada em qualquer aplicativo front-end, não apenas no React.

E o Redux?

- O conjunto de ferramentas Redux é muito semelhante ao que acabamos de ver com useReducer, mas com mais algumas coisas. Existem três blocos de construção principais no Redux:
 - **Uma loja** - um objeto que contém os dados de estado do aplicativo
 - **Um redutor** - uma função que retorna alguns dados de estado, acionados por um tipo de ação
 - **Uma ação** - um objeto que informa ao redutor como alterar o estado. Ele deve conter uma propriedade de tipo.

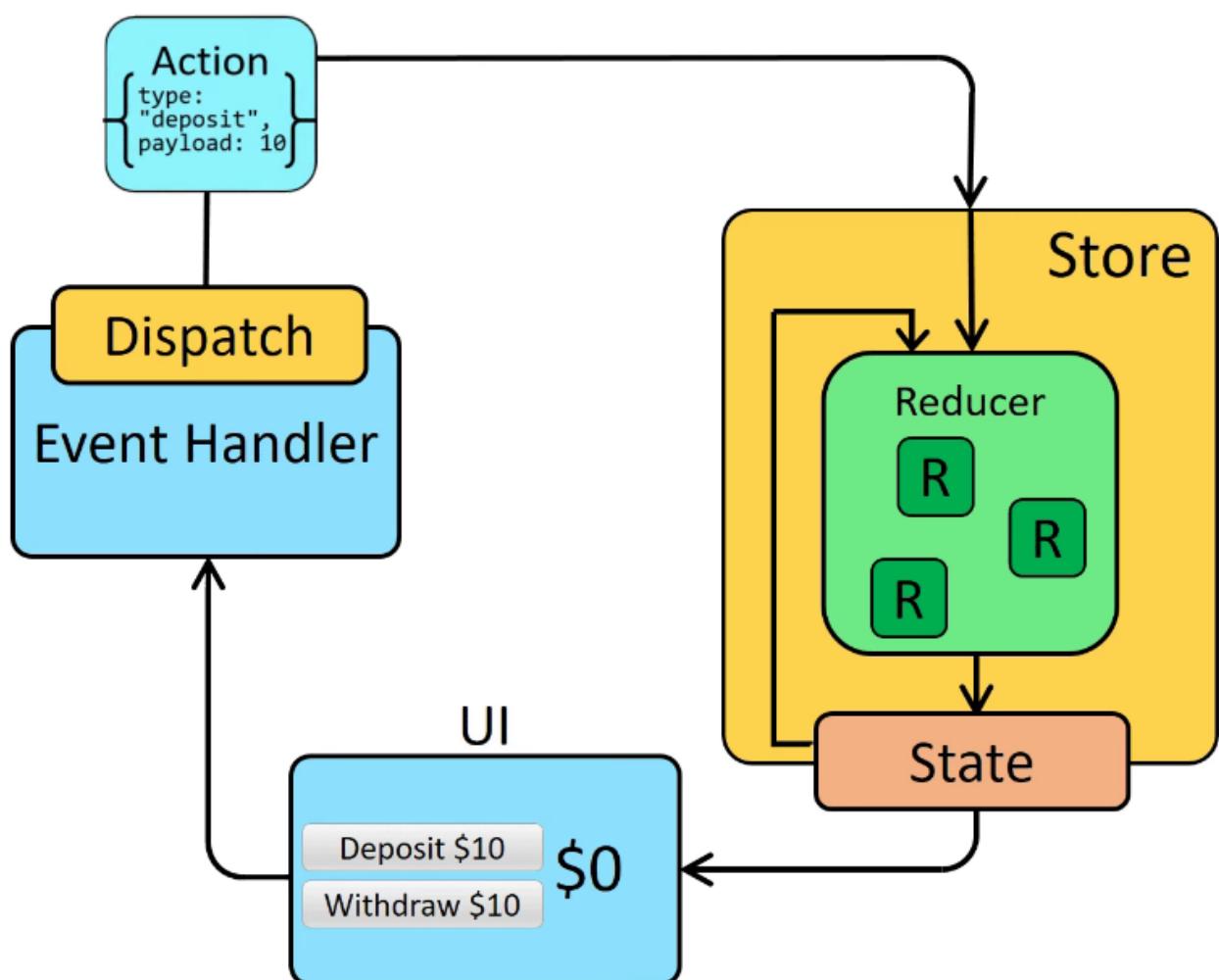
E o Redux?

- O conjunto de ferramentas Redux é muito semelhante ao que acabamos de ver com useReducer, mas com mais algumas coisas. Existem três blocos de construção principais no Redux:
 - **Uma loja** - um objeto que contém os dados de estado do aplicativo
 - **Um redutor** - uma função que retorna alguns dados de estado, acionados por um tipo de ação
 - **Uma ação** - um objeto que informa ao redutor como alterar o estado. Ele deve conter uma propriedade de tipo.



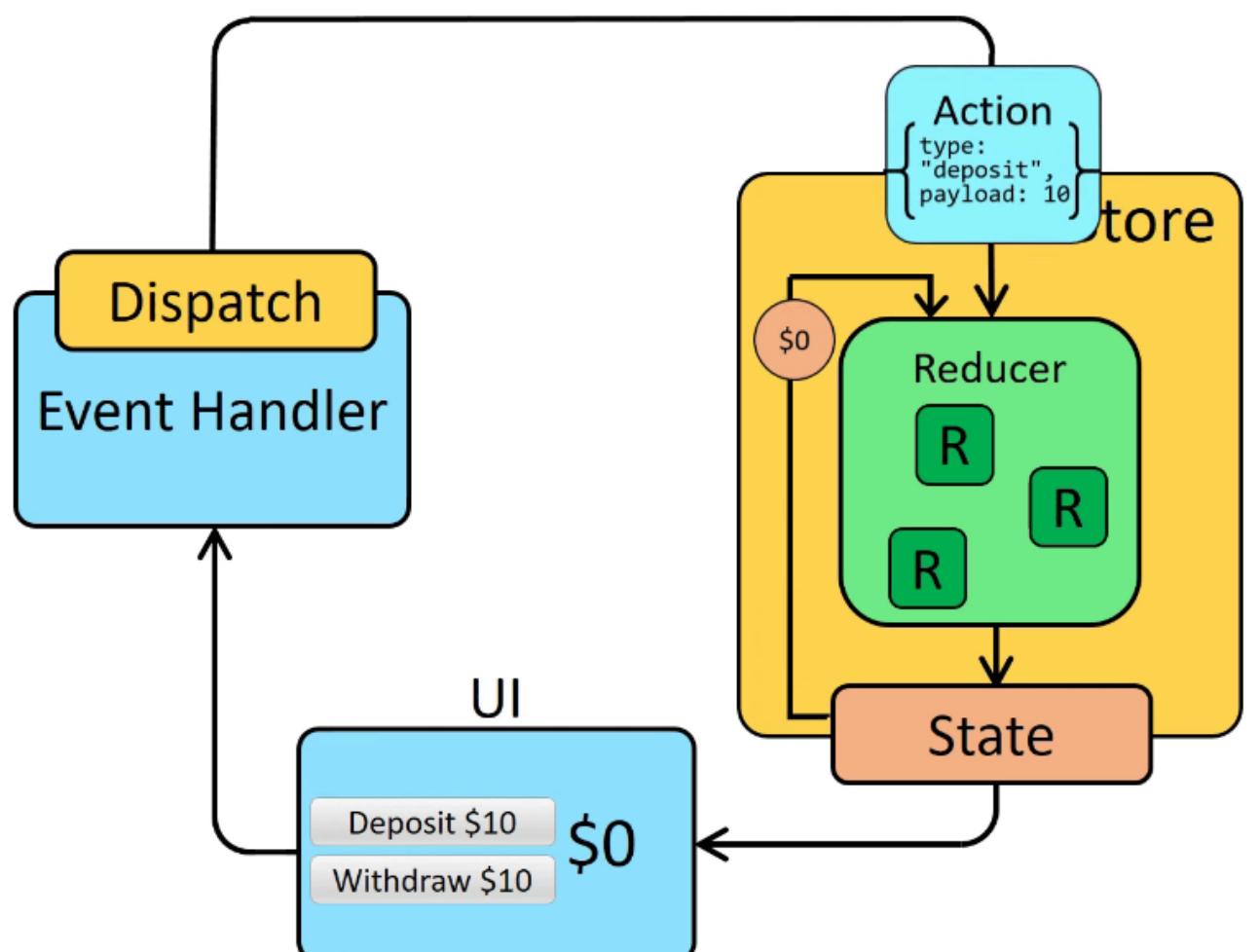
E o Redux?

- O conjunto de ferramentas Redux é muito semelhante ao que acabamos de ver com useReducer, mas com mais algumas coisas. Existem três blocos de construção principais no Redux:
 - **Uma loja** - um objeto que contém os dados de estado do aplicativo
 - **Um redutor** - uma função que retorna alguns dados de estado, acionados por um tipo de ação
 - **Uma ação** - um objeto que informa ao redutor como alterar o estado. Ele deve conter uma propriedade de tipo.



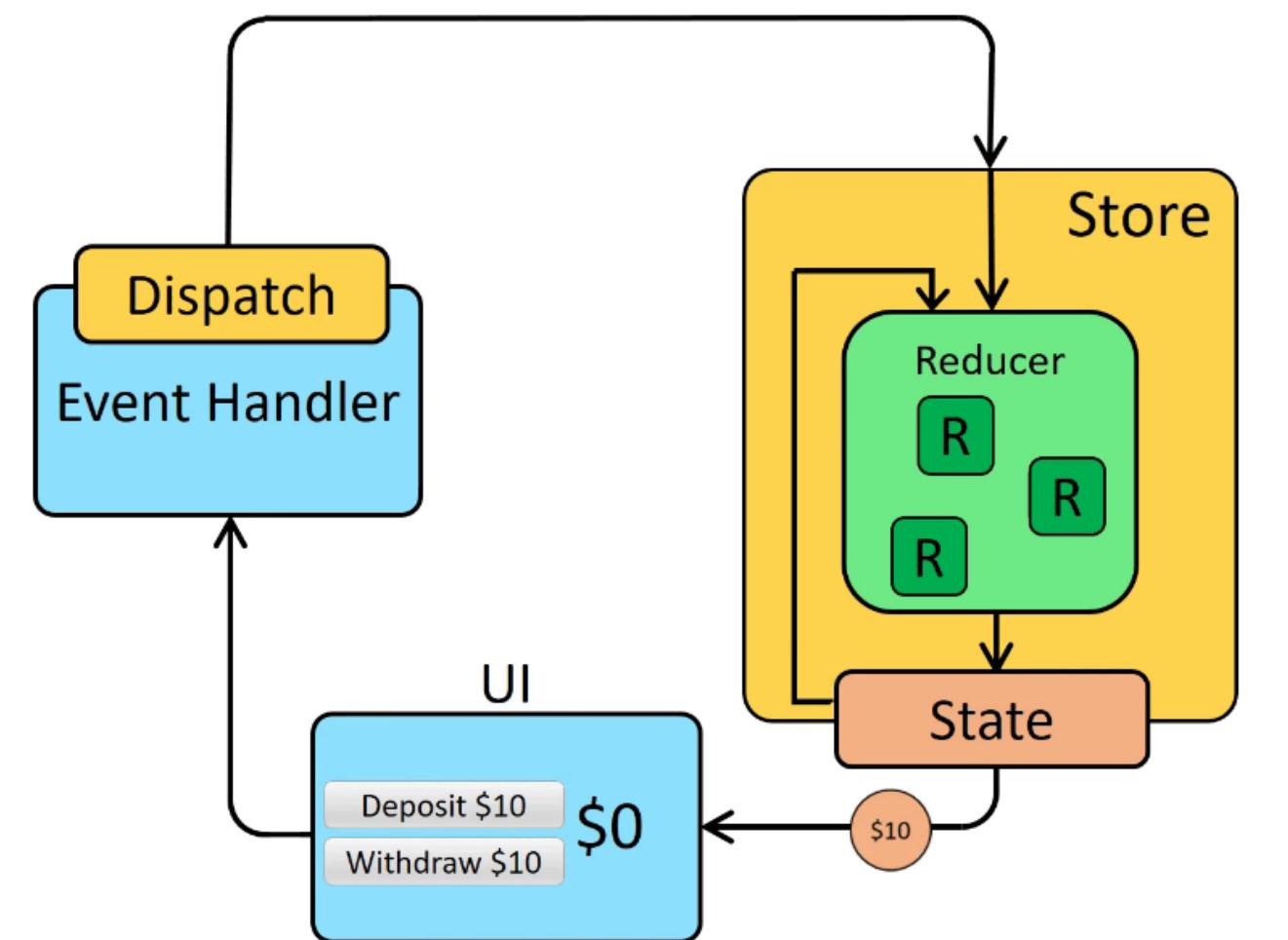
E o Redux?

- O conjunto de ferramentas Redux é muito semelhante ao que acabamos de ver com useReducer, mas com mais algumas coisas. Existem três blocos de construção principais no Redux:
 - **Uma loja** - um objeto que contém os dados de estado do aplicativo
 - **Um redutor** - uma função que retorna alguns dados de estado, acionados por um tipo de ação
 - **Uma ação** - um objeto que informa ao redutor como alterar o estado. Ele deve conter uma propriedade de tipo.

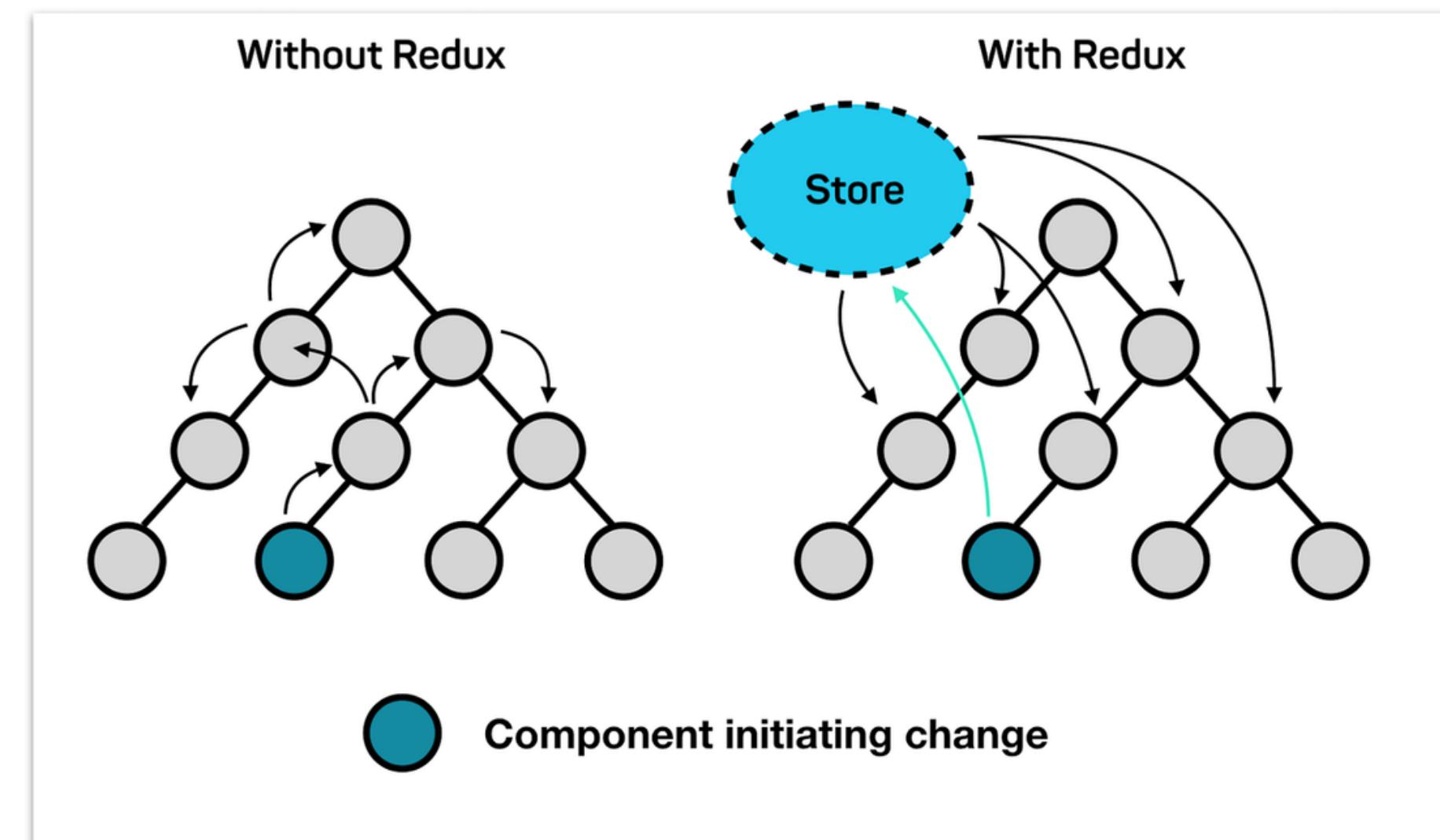


E o Redux?

- O conjunto de ferramentas Redux é muito semelhante ao que acabamos de ver com useReducer, mas com mais algumas coisas. Existem três blocos de construção principais no Redux:
 - **Uma loja** - um objeto que contém os dados de estado do aplicativo
 - **Um redutor** - uma função que retorna alguns dados de estado, acionados por um tipo de ação
 - **Uma ação** - um objeto que informa ao redutor como alterar o estado. Ele deve conter uma propriedade de tipo.



Qual é a diferença entre useState e Redux?



Vamos implementar o Redux

Criando o Counter Slice

src/redux/slices/count.slice.ts

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

export const countSlice = createSlice({
  name: "countSlice",
  initialState: {
    value: 0,
  },
  reducers: {
    increment(state) {
      state.value += 1;
    },
  },
});

export const { increment } = countSlice.actions;
export default countSlice.reducer;
```

Criando o Store

src/redux/store.ts

```
import { configureStore } from "@reduxjs/toolkit";
import countReducer from "./slices/count.slice";

export const store = configureStore({
  reducer: {
    count: countReducer,
  },
});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

Importando o Store

index.tsx

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

const root = ReactDOM.createRoot(
  document.getElementById("root") as HTMLElement
);
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);

reportWebVitals();
```

Utilizando o Redux

App.tsx

```
import "./App.css";
import { useDispatch, useSelector } from "react-redux";
import { increment } from "./redux/slices/count.slice";
import { RootState } from "./redux/store";

function App() {
  const dispatch = useDispatch();
  const count = useSelector((state: RootState) => state.count);

  return (
    <div className="App">
      <p>{count.value}</p>
      <button onClick={() => dispatch(increment())}> +1 </button>
    </div>
  );
}

export default App;
```

Atividade

1. Criar novos Reducers para elevar o valor ao quadrado e decrementar

Utilizando arrays - Criando Produto Slice

src/redux/slices/produto.slice.ts

```
● ● ●

import { createSlice, PayloadAction } from "@reduxjs/toolkit";

interface Produto {
  id: number;
  nome: string;
  preco: number;
  estoque: number;
  createdAt: string;
  updatedAt: string;
}

interface ProdutosState {
  produtos: Produto[];
}

const initialState: ProdutosState = {
  produtos: [
    {
      id: 0,
      nome: "Maça",
      preco: 1.30,
      estoque: 10,
      createdAt: "",
      updatedAt: ""
    },
  ],
};

const produtoSlice = createSlice({
  name: "produtos",
  initialState,
  reducers: {
    addProduto: (state, action: PayloadAction<Produto>) => {
      state.produtos.push(action.payload);
    },
    removeProduto: (state, action: PayloadAction<number>) => {
      state.produtos = state.produtos.filter(
        (produto) => produto.id !== action.payload
      );
    },
  },
});

export const { addProduto, removeProduto } =
  produtoSlice.reducer;
```

Inserindo no Store

src/redux/store.ts

```
import { configureStore } from "@reduxjs/toolkit";
import countReducer from "./slices/count.slice";
import produtoReducer from "./slices/produtos.slice";

export const store = configureStore({
  reducer: {
    count: countReducer,
    produto: produtoReducer,
  },
});

export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

Criando Componente para Listar Produtos

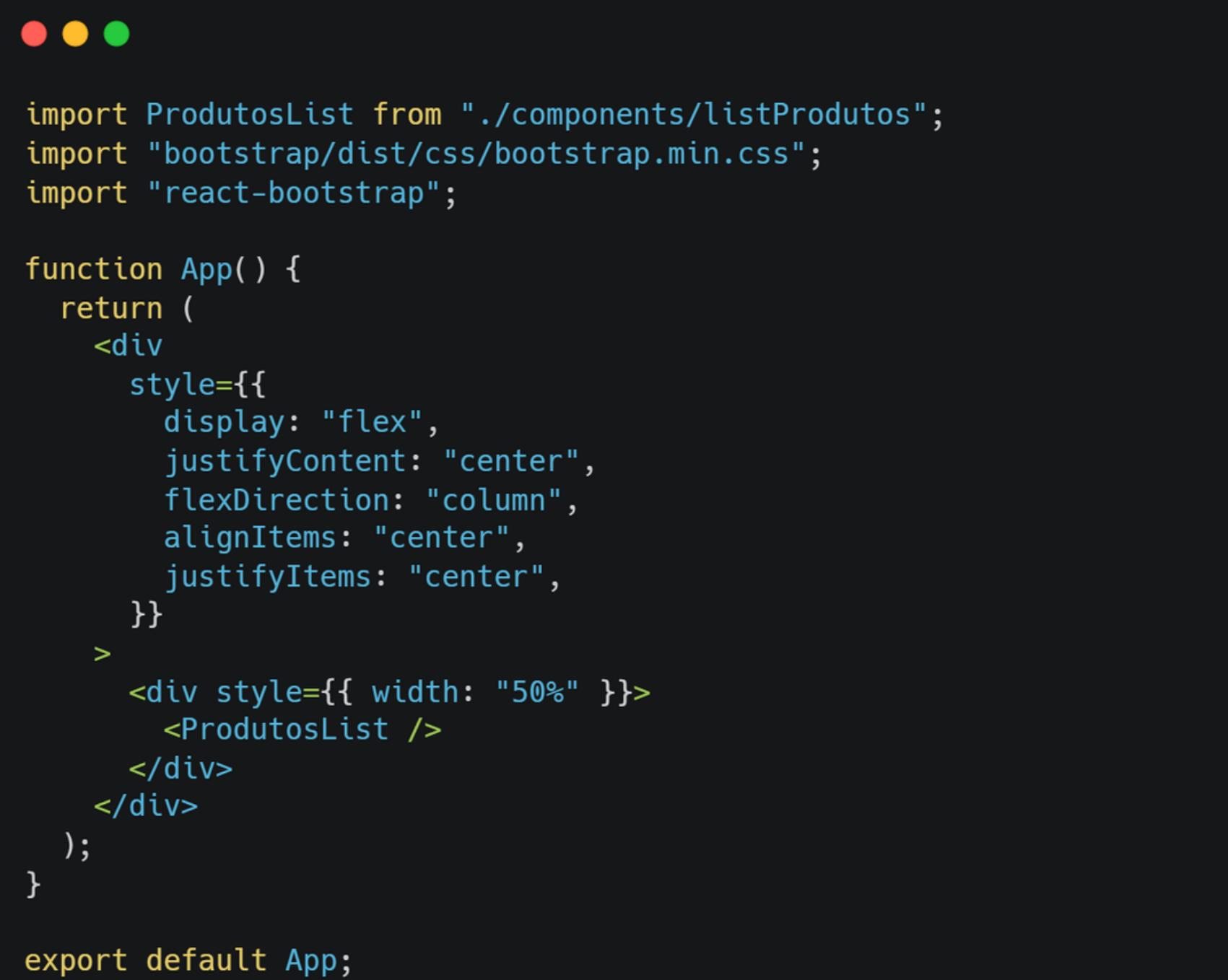
src/components/listProdutos.tsx

```
● ● ●  
  
import "bootstrap/dist/css/bootstrap.min.css";  
import { useSelector } from "react-redux/es/hooks/useSelector";  
import { RootState } from "../redux/store";  
  
export default function ProdutosList() {  
  const produtos = useSelector((state: RootState) => state.produto.produtos);  
  
  return (  
    <div>  
      <table className="table">  
        <thead>  
          <tr>  
            <th scope="col">#</th>  
            <th scope="col">Nome</th>  
          </tr>  
        </thead>  
        <tbody>  
          {produtos.map((produto) => {  
            return (  
              <tr>  
                <th scope="row">{produto.id}</th>  
                <td>{produto.nome}</td>  
              </tr>  
            );  
          })}  
        </tbody>  
      </table>  
    </div>  
  );  
}
```

npm install reactstrap react react-dom
npm install --save bootstrap
npm i reactstrap
npm i react-bootstrap

Chamando o Componente

src/App.tsx



```
● ● ●

import ProdutosList from "./components/listProdutos";
import "bootstrap/dist/css/bootstrap.min.css";
import "react-bootstrap";

function App() {
  return (
    <div
      style={{
        display: "flex",
        justifyContent: "center",
        flexDirection: "column",
        alignItems: "center",
        justifyItems: "center",
      }}
    >
      <div style={{ width: "50%" }}>
        <ProdutosList />
      </div>
    </div>
  );
}

export default App;
```

Form de inserção de Produtos

```
import { useState } from "react";
import { useDispatch } from "react-redux";
import { addProduto } from "../redux/slices/produtos.slice";

import "bootstrap/dist/css/bootstrap.min.css";
import "react-bootstrap";

export default function FormularioProduto() {
  const dispatch = useDispatch();

  const [inputProduto, setInputProduto] = useState({
    id: 0,
    nome: "",
    preco: 0,
    estoque: 0,
    createdat: "",
    updatedAt: ""
  });

  const handleInput = (e: any) => {
    setInputProduto({ ...inputProduto, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e: any) => {
    e.preventDefault();
    dispatch(addProduto(inputProduto));
  };

  return (
    <div style={{ display: "flex", flexDirection: "column", alignItems: "center" }}>
      <h3 className="mt-3">Inserir Novo Produto</h3>
      <form onSubmit={handleSubmit}>
        <div className="row mb-3">
          <label className="col-sm-3 col-form-label">Nome</label>
          <div className="col-md-8">
            <input
              type="text"
              className="form-control"
              name="nome"
              value={inputProduto.nome}
              onChange={handleInput}
            />
          </div>
        </div>

        <div className="row mb-3">
          <label className="col-sm-3 col-form-label">Preço</label>
          <div className="col-md-8">
            <input
              type="text"
              className="form-control"
              name="preco"
              value={inputProduto.preco}
              onChange={handleInput}
            />
          </div>
        </div>

        <div className="row mb-3">
          <label className="col-sm-3 col-form-label">Estoque</label>
          <div className="col-md-8">
            <input
              type="text"
              className="form-control"
              name="estoque"
              value={inputProduto.estoque}
              onChange={handleInput}
            />
          </div>
        </div>

        <div className="row mb-3">
          <label className="col-sm-3 col-form-label"></label>
          <div className="col-md-1">
            <button type="submit" className="btn btn-primary btn-lg">
              Submit
            </button>
          </div>
        </div>
      </form>
    </div>
  );
}
```

src/components/formProduto.tsx

Chamando o Componente

App.tsx

```
● ● ●

import ProdutosList from "./components/listProdutos";
import "bootstrap/dist/css/bootstrap.min.css";
import "react-bootstrap";
import FormularioProduto from "./components/formProduto";

function App() {
  return (
    <div
      style={{
        display: "flex",
        justifyContent: "center",
        flexDirection: "column",
        alignItems: "center",
        justifyItems: "center",
      }}
    >
      <div style={{ width: "50%" }}>
        <FormularioProduto />
        <ProdutosList />
      </div>
    </div>
  );
}

export default App;
```

Atividade

- 1. Inserir os outros campos na tabela**
- 2. Chamar os dispach de remover o Produto da Lista através de um botão**
- 3. Incrementar o ID automaticamente**

Persistindo os dados

npm install redux-persist redux-persist-webstorage

src/redux/persistConfig.ts

```
● ● ●

import { combineReducers, configureStore } from "@reduxjs/toolkit";
import countReducer from "./slices/count.slice";
import produtoReducer from "./slices/produtos.slice";
import storage from "redux-persist/lib/storage";
import { persistReducer, persistStore } from "redux-persist";

const rootReducer = combineReducers({
  count: countReducer,
  produto: produtoReducer,
});

const persistConfig = {
  key: "root",
  storage,
};

const persistedReducer = persistReducer(persistConfig, rootReducer);

export const store = configureStore({
  reducer: persistedReducer,
});

export const persistor = persistStore(store);
export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

Persistindo os dados

index.tsx

```
● ● ●

import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { Provider } from "react-redux";
import { persistor, store } from "./redux/store";
import { PersistGate } from "redux-persist/integration/react";

const root = ReactDOM.createRoot(
  document.getElementById("root") as HTMLElement
);
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <PersistGate loading={null} persistor={persistor}>
        <App />
      </PersistGate>
    </Provider>
  </React.StrictMode>
);

reportWebVitals();
```

ToDoList - Ativida

React Todo App

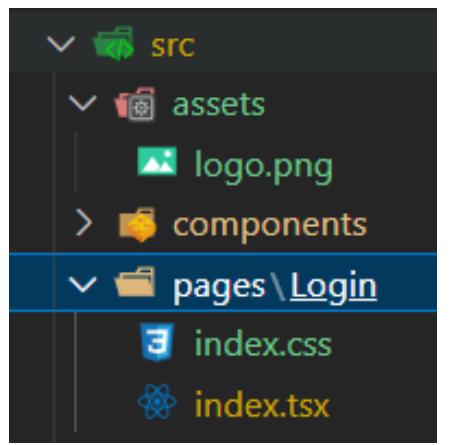
Enter Something	Add
HTML Tutorial	
CSS Tutorial	
JavaScript Tutorial	
jQuery Tutorial	
PHP Tutorial	
Node.js Tutorial	
React Js Tutorial	

React Todo App

Enter Something	Add
HTML Tutorial	
CSS Tutorial	
JavaScript Tutorial	
jQuery Tutorial	
PHP Tutorial	
Node.js Tutorial	
React Js Tutorial	

Login

Criar pasta de Login



index.tsx

```
import { useState } from "react";
import { Form, Image, Button } from "react-bootstrap";
import logo from "../../assets/logo.png";

import "bootstrap/dist/css/bootstrap.min.css";
import "./index.css";

export default function Login() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  return (
    <Form style={{ width: "300px" }} className="container">
      <div style={{ margin: "0 auto" }}>
        <Image className="imageBorder" src={logo} />
      </div>

      <input
        placeholder="Digite aqui seu email"
        onChange={(e) => {
          setEmail(e.target.value);
        }}
      />

      <input
        type="password"
        placeholder="Digite aqui sua senha"
        onChange={(e) => {
          setPassword(e.target.value);
        }}
      />
      <Button>Login</Button>
    </Form>
  );
}
```

index.css

```
.container {
  min-height: 100vh;
  align-items: 'center';
  display: flex;
  gap: 30px;
  justify-content: center;
  flex-direction: column;
}

.imageBorder {
  width: 150px;
  height: 150px;
  border-radius: 50%;
}
```

npm i react-router-dom

index.tsx

```
const router = createBrowserRouter([
  { path: "/", element: <Login /> },
  { path: "/home", element: <App /> },
]);

root.render(
  <React.StrictMode>
    <Provider store={store}>
      <PersistGate loading={true} persistor={persistor}>
        <RouterProvider router={router} />
      </PersistGate>
    </Provider>
  </React.StrictMode>
);
```

Slice de Login

api.slice.login.ts

```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import axios, { AxiosResponse } from "axios";

interface ApiState {
  loading: boolean;
  data: object;
  error: string;
  isSucess: boolean;
}

const initialState: ApiState = {
  loading: false,
  data: {},
  error: "",
  isSucess: false,
};

export const doLogin = createAsyncThunk(
  "api/login",
  async (dataLogin: object) => {
    const response: AxiosResponse = await axios.post(
      "http://localhost:3333/v1/login",
      dataLogin
    );

    return { payload: response.data, status: response.status };
  }
);

const apiLoginSlice = createSlice({
  name: "apiLogin",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(doLogin.pending, (state) => {
        state.loading = true;
        state.error = "";
        state.isSucess = false;
      })
      .addCase(doLogin.fulfilled, (state, action) => {
        state.loading = false;
        state.data = action.payload;

        if (action.payload.status === 200) {
          state.isSucess = true;
        }
      })
      .addCase(doLogin.rejected, (state, action) => {
        state.loading = false;
        state.error = action.error.message ?? "";
        state.isSucess = false;
      });
  },
};

export const { reducer: apiLoginReducer } = apiLoginSlice;
export default apiLoginSlice;
```

store.ts

```
import { combineReducers, configureStore } from "@reduxjs/toolkit";
import countReducer from "./slices/count.slice";
import { apiLoginReducer } from "./slices/api.slice.login";

import storage from "redux-persist/lib/storage";
import {
  persistReducer,
  persistStore,
  FLUSH,
  REHYDRATE,
  PAUSE,
  PERSIST,
  PURGE,
  REGISTER,
} from "redux-persist";

const rootReducer = combineReducers({
  count: countReducer,
  apiLogin: apiLoginReducer,
});

const persistConfig = {
  key: "root",
  storage,
  blacklist: ["apiLogin"],
};

const persistedReducer = persistReducer(persistConfig, rootReducer);

export const store = configureStore({
  reducer: persistedReducer,
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({
      serializableCheck: {
        ignoredActions: [FLUSH, REHYDRATE, PAUSE, PERSIST, PURGE, REGISTER],
      },
    }),
};

export const persistor = persistStore(store);
export type RootState = ReturnType<typeof store.getState>;
export type AppDispatch = typeof store.dispatch;
```

login/index.tsx

```
import { useEffect, useState } from "react";
import { Form, Image, Button } from "react-bootstrap";
import logo from "../../assets/logo.png";
import { useDispatch, useSelector } from "react-redux";

import "bootstrap/dist/css/bootstrap.min.css";
import "./index.css";

import { doLogin } from "../../../../redux/slices/api.slice.login";
import { AppDispatch, RootState } from "../../../../redux/store";
import { useNavigate } from "react-router-dom";

export default function Login() {
  const { error, loading, isSucess } = useSelector((state: RootState) => state.apiLogin);
  const dispatch = useDispatch<AppDispatch>();
  const navigate = useNavigate();

  const [email, SetEmail] = useState("");
  const [password, SetPassword] = useState("");

  useEffect(() => {
    if (isSucess) {
      navigate("/home");
    }
  }, [isSucess]);

  const TryLogin = () => {
    dispatch(
      doLogin({
        email: email,
        senha: password,
      })
    );
  };

  return (
    <Form style={{ width: "300px" }} className="container">
      <div style={{ margin: "0 auto" }}>
        <Image className="imageBorder" src={logo} />
      </div>
      <input
        placeholder="Digite aqui seu email"
        onChange={(e) => {
          SetEmail(e.target.value);
        }}
      />
      <input
        type="password"
        placeholder="Digite aqui sua senha"
        onChange={(e) => {
          SetPassword(e.target.value);
        }}
      />
      {loading ? (
        "Loading..."
      ) : (
        <Button onClick={() => TryLogin()}>Login</Button>
      )}
      {error}
    </Form>
  );
}
```

Consumir dados de uma API de Produtos

api.slice.producto.ts

```
import { createSlice, createAsyncThunk, PayloadAction } from "@reduxjs/toolkit";
import axios, { AxiosResponse } from "axios";

interface Produto {
  id: number;
  nome: string;
  preco: number;
  estoque: number;
  createdAt: string;
  updatedAt: string;
}

interface ApiState {
  loading: boolean;
  produtos: Produto[];
  error: string;
}

const initialState: ApiState = {
  loading: false,
  produtos: [],
  error: "",
};

export const fetchProdutos = createAsyncThunk<Produto[]>(
  "api/get/produtos",
  async () => {
    const wait = (ms: number) => new Promise((res) => setTimeout(res, ms));
    await wait(1000);

    const response: AxiosResponse<Produto[]> = await axios.get(
      "http://localhost:3333/v1/produto"
    );
    return response.data;
  }
);

const apiProdutoSlice = createSlice({
  name: "apiProduto",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(fetchProdutos.pending, (state) => {
        state.loading = true;
        state.error = "";
      })
      .addCase(
        fetchProdutos.fulfilled,
        (state, action: PayloadAction<Produto[]>) => {
          state.loading = false;
          state.produtos = action.payload;
        }
      )
      .addCase(fetchProdutos.rejected, (state, action: PayloadAction<any>) => {
        state.loading = false;
        state.error = action.payload;
      });
  },
});

export const { reducer: apiProdutoReducer } = apiProdutoSlice;
export default apiProdutoSlice;
```

store.tsx

```
/ const rootReducer = combineReducers({
  count: countReducer,
  apiLogin: apiLoginReducer,
  apiProduto: apiProdutoReducer,
});

/ const persistConfig = {
  key: "root",
  storage,
  blacklist: ["apiLogin", "apiProduto"],
};
```

App.tsx

```
import "bootstrap/dist/css/bootstrap.min.css";
import "react-bootstrap";
import FormularioProduto from "./components/formProduto";
import { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { AppDispatch, RootState } from "./redux/store";
import { fetchProdutos } from "./redux/slices/api.slice.produtos";
import ProdutosList from "./components/listProdutos";

function App() {
  const { error, loading } = useSelector(
    (state: RootState) => state.apiProduto
  );

  const dispatch = useDispatch<AppDispatch>();

  useEffect(() => {
    dispatch(fetchProdutos());
  }, []);

  return (
    <div
      style={{
        display: "flex",
        justifyContent: "center",
        flexDirection: "column",
        alignItems: "center",
        justifyItems: "center",
      }}
    >
      {loading ? "Loading..." : (
        <div style={{ width: "50%" }}>
          <FormularioProduto />
          <ProdutosList />
        </div>
      )}
    </div>
  );
}

export default App;
```

Inserindo dados em uma API de Produtos

api.slice.producto.ts

```
export const addProduto = createAsyncThunk(
  "api/post/produto",
  async (produto: Produto) => {
    const wait = (ms: number) => new Promise((res) => setTimeout(res, ms));
    await wait(1000);

    const response: AxiosResponse<Produto> = await axios.post(
      "http://localhost:3333/v1/produto",
      produto,
      { withCredentials: true }
    );

    return response.data;
  }
);

.addCase(
  addProduto.fulfilled,
  (state, action: PayloadAction<Produto>) => {
    state.loading = false;
    state.produtos.push(action.payload);
  }
);
```

App.tsx

```
const dispatch = useDispatch<AppDispatch>();

const [inputProduto, SetProduto] = useState({
  nome: "",
  preco: 0,
  estoque: 0,
});

const handleSubmit = (e: any) => {
  e.preventDefault();
  dispatch(addProduto(inputProduto));
};
```

Removendo produtos da Lista

api.slice.produto.ts

```
export const removeProduto = createAsyncThunk(
  "api/delete/produto",
  async (id: string) => {
    const wait = (ms: number) => new Promise((res) => setTimeout(res, ms));
    await wait(1000);

    const response: AxiosResponse<Produto> = await axios.delete(
      `http://localhost:3333/v1/produto/${id}`
    );

    return { payload: response.data, removedId: id };
};
```

```
.addCase(removeProduto.fulfilled, (state, action) => {
  state.loading = false;
  state.produtos = state.produtos.filter(
    (produto) => produto.id !== action.payload.removedId
  );
});
```

ListProdutos

```
export default function ProdutosList() {
  const { produtos } = useSelector((state: RootState) => state.apiProduto);
  const dispatch = useDispatch<AppDispatch>();

  function RemoveProduto(id: any) {
    dispatch(removeProduto(id));
  }
}
```

```
<th scope="col">Ação Remover</th>
```

```
<td>
  <Button onClick={() => RemoveProduto(produto.id)}>
    Remover
  </Button>
</td>
```

Tipo de Usuario

```
api.slice.login.ts
state.isAdmin = action.payload.payload.isAdmin;
if (action.payload.status === 200) {
  state.isSucess = true;
}
```

```
<th scope="col">Estoque</th>
{isAdmin ? (
  <th scope="col">Remover Produto</th>
) : (
  <th scope="col">Inserir Carrinho</th>
)}
```

```
<td>
{isAdmin ? (
  <Button onClick={() => RemoveProduto(produto.id)}>
    Remover
  </Button>
) : (
  <Button onClick={() => RemoveProduto(produto.id)}>
    Adicionar
  </Button>
)}
</td>
```

Tipo de Usuario

```
api.slice.login.ts
state.isAdmin = action.payload.payload.isAdmin;
if (action.payload.status === 200) {
  state.isSucess = true;
}
```

```
<th scope="col">Estoque</th>
{isAdmin ? (
  <th scope="col">Remover Produto</th>
) : (
  <th scope="col">Inserir Carrinho</th>
)}
```

```
<td>
{isAdmin ? (
  <Button onClick={() => RemoveProduto(produto.id)}>
    Remover
  </Button>
) : (
  <Button onClick={() => RemoveProduto(produto.id)}>
    Adicionar
  </Button>
)}
</td>
```



IMAGE	PRODUCT	PRICE	QUANTITY	ACTIONS	TOTAL PRICE
	Grand Theft Auto V	\$ 21.99	2		\$ 43.98
	Spider-Man Miles Morales	\$ 29.99	3		\$ 89.97
	Flight Simulator 2020	\$ 59.99	5		\$ 299.95
	Minecraft	\$ 49.99	1		\$ 49.99

Grand Total: \$ 483.89

Obrigado