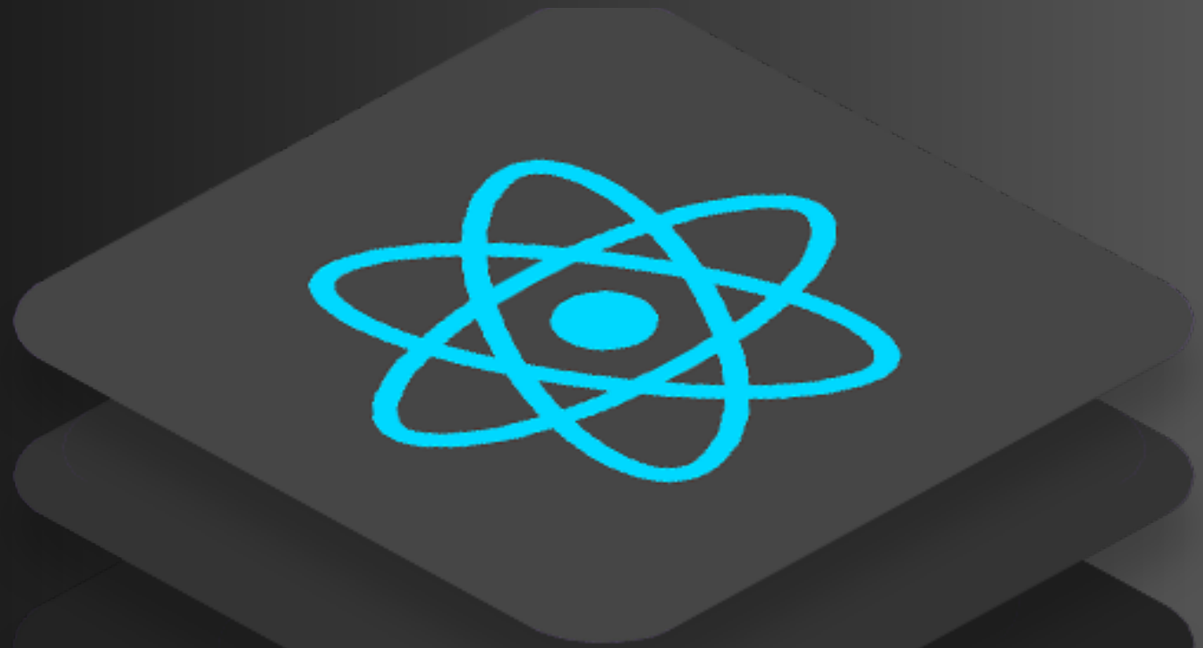

React

Frontend

Roteamento de Páginas em React

O React é por definição uma Single Page Application (SPA) - Aplicação de página única. O que isso significa? A aplicação será renderizada em uma única página.

Além de aplicações SPA existem diversas outras:

Tipo de aplicação	Características
SPA	Renderização do lado do cliente
MPA	Renderização do lado do servidor
Híbrida	Renderização do lado do servidor e do cliente
Móvel	Renderização do lado do cliente

MPA

As aplicações MPA (Multi Page Application) são aplicações web tradicionais. Elas são compostas por várias páginas HTML que são carregadas dinamicamente pelo navegador. Isso oferece mais flexibilidade para o desenvolvedor, pois permite que ele crie aplicações mais complexas.

Aplicações híbridas

As aplicações híbridas são uma combinação de aplicações web e aplicativos móveis. Elas são criadas usando uma linguagem de programação web, como JavaScript ou HTML, mas são empacotadas em um aplicativo nativo para dispositivos móveis. Isso permite que as aplicações ofereçam uma experiência de usuário mais nativa, pois aproveitam as capacidades dos dispositivos móveis. (Ionic, ReactNative etc)

Aplicações móveis

As aplicações móveis são aplicativos que são desenvolvidos especificamente para dispositivos móveis, como smartphones e tablets. Elas são criadas usando uma linguagem de programação nativa para o dispositivo, como Swift, Kotlin ou Objective-C. Isso permite que as aplicações ofereçam uma experiência de usuário mais otimizada para os dispositivos móveis.

Como fazer paginação em uma SPA?

Existem bibliotecas para paginação de SPA, em especial para React existem:

React Router DOM - É uma biblioteca de roteamento completa e fácil de usar com grande comunidade e extensa documentação. Uma das grandes vantagens é possibilidade de utilização no gerenciamento de rotas em aplicações de grande porte

Reach Router - É uma biblioteca biblioteca de roteamento leve e minimalista. Utilizada em aplicações de pequeno porte.

React Router DOM

É uma biblioteca JavaScript que permite gerenciar o roteamento em aplicações React.
Por que usar o React Router DOM?
O React Router DOM oferece uma série de vantagens, incluindo:

- **Facilidade de uso:** O React Router DOM é fácil de aprender e usar, com uma API simples e intuitiva.

- **Flexibilidade:** O React Router DOM oferece uma ampla gama de funcionalidades, permitindo que você crie aplicações complexas com facilidade.

- **Compatibilidade:** O React Router DOM é compatível com todas as versões mais recentes do React.

Importação e estrutura básica

```jsx
import {
  BrowserRouter,
  Link,
  Route,
  Routes,
} from "react-router-dom";

export default function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/blog/*" element={<BlogApp />} />
        <Route path="/users/*" element={<UserApp />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Instalação

```
npm install react-router-dom
```

BrowserRouter

O BrowserRouter é responsável pelas seguintes tarefas:

- Definir o elemento raiz da aplicação.
- Gerenciar o histórico de navegação.
- Resolver conflitos de rotas.

Routes

O Routes é responsável pelas seguintes tarefas:

- Agrupar rotas.
- Gerenciar o roteamento.

Route

O Route é responsável pelas seguintes tarefas:

- Definir uma rota específica.
- Determinar se a rota corresponde ao caminho atual da aplicação.
- Renderizar o componente associado à rota.

Navegação entre telas

Link

A <Link> é um elemento que permite ao usuário navegar para outra página clicando ou tocando nela.

```
import * as React from "react";
import { Link } from "react-router-dom";

function UsersIndexPage({ users }) {
  return (
    <div>
      <h1>Users</h1>
      <ul>
        {users.map((user) => (
          <li key={user.id}>
            <Link to={user.id}>{user.name}</Link>
          </li>
        ))}
      </ul>
    </div>
  );
}
```

Hook useNavigate

O useNavigate hook retorna uma função que permite navegar programaticamente, por exemplo, em um efeito:

```
import { useNavigate } from "react-router-dom";

function useLogoutTimer() {
  const userIsInactive = useFakeInactiveUser();
  const navigate = useNavigate();

  useEffect(() => {
    if (userIsInactive) {
      fake.logout();
      navigate("/session-timed-out");
    }
  }, [userIsInactive]);
}
```

Parâmetros de rota

Os parâmetros de rota são uma maneira de tornar as rotas dinâmicas e flexíveis. Eles permitem que você passe informações específicas para uma rota, permitindo que a mesma rota seja reutilizada com diferentes dados.

```
import { BrowserRouter as Router, Route } from 'react-router-dom';
import ItemDetail from './ItemDetail';

const App = () => {
  return (
    <Router>
      <Route path="/item/:id" element={<ItemDetail />} />
    </Router>
  );
};
```

useParams()

O hook useParams é utilizado para acessar os params da rota

```jsx
import { useParams } from 'react-router-dom';

const ItemDetail = () => {
  // Use o hook useParams para acessar os parâmetros de rota
  const { id } = useParams();

  return (
    <div>
      <h2>Detalhes do Item {id}</h2>
      {/* Restante do código... */}
    </div>
  );
};
```

Ao acessar a rota item/1 ou item/2, pode-se utilizar o hook useParams() para acessar o id apontado na rota (no caso, id pode ser 1 ou 2, respectivamente)

Aninhamento de Rotas

O componente Route permite ainda o aninhamento de rotas de modo a criar camadas de acesso, permitindo a modularização das rotas

```
<>
    <Route path="/" element={<HomePage />} />
    <Route path="/products" element={<ProductsPage />}>
        <Route path=":id" element={<ProductDetail />} />
    </Route>
    <Route path="/cart" element={<CartPage />} >
        <Route path="checkout" element={<CheckOutPage />} />
    </Route>

</>
```

Rotas Autenticadas

No intuito de garantir a segurança da aplicação, pode-se aplicar a renderização condicional de rotas

```
    const[userAuthenticated, setUserAuthenticated] = useState(true)

    return (
        <>
            <Route path="/dashboard" element={<PrivateRoute element={<Dashboard />}
            authenticated={userAuthenticated} />} />
        </>
    )
}
const PrivateRoute = ({ element, authenticated }) => {
    return authenticated ? element : <Navigate to="/login" />;
};
```

Vamos à prática