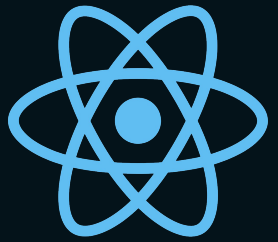
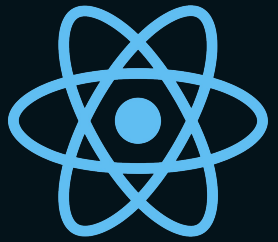
Como vai funcionar o curso?

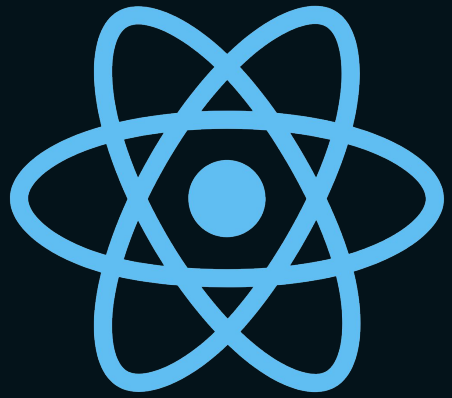
- Cada exercício tem duas aulas: apresentação do problema e solução;
- As instruções devem ser seguidas para a realização do exercício;
- Faça pesquisas para relembrar conceitos que não estão tão claros ou que você não conhece;
- Mesmo não conseguindo resolver sozinho, veja o vídeo de solução e faça o código comigo;
- Depois implemente outro exercício semelhante para fixar o conteúdo;



Repositório do curso

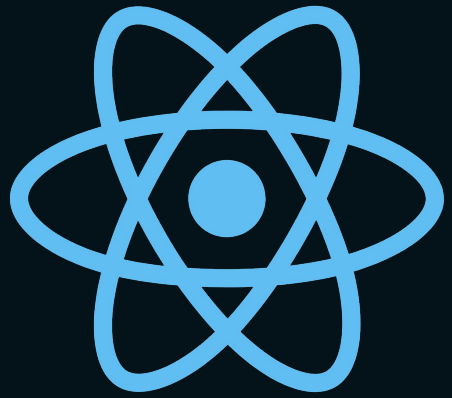
- Todos os arquivos podem ser encontrados em: https://github.com/matheusbattisti/desafios_react
- Lá você pode baixar tudo ou comparar com o seu código;
- Assim você terá todo o código como material de apoio, além das aulas;
- Vamos começar? =)





Introdução

Conclusão da seção

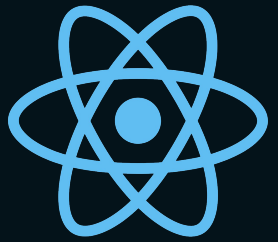


Componentes e Props

introdução da seção

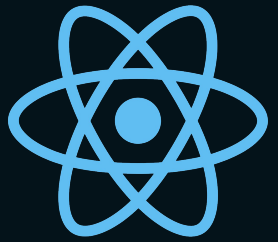
Ex 1: Crie um componente simples

- Crie um componente chamado HelloWorld;
- Este componente deve renderizar um parágrafo;
- Insira um texto neste parágrafo;



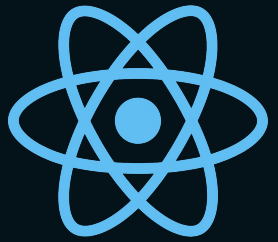
Ex 2: Usando Props

- Crie um novo componente;
- Este componente deve receber uma prop chamada name;
- Renderize um texto no componente, contendo a prop que foi enviada;



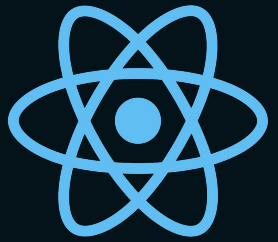
Ex 3: Múltiplas props

- Crie um novo componente;
- Este componente deve receber pelo menos três pops, utilize tipos de dados diferentes;
- Exiba os valores das props no componente;



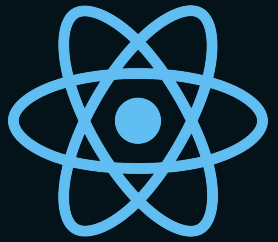
Ex 4: Composição de Componentes

- Crie um novo Componente;
- Este componente deve aceitar uma prop chamada members;
- Em members teremos um array de objetos;
- Defina este array com as propriedades iguais as props do componente do exercício anterior;
- Utilizando os componentes em conjunto, renderize todos os itens do array com um map;



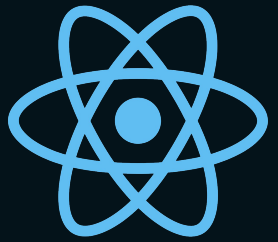
Ex 5: Componente de Classe

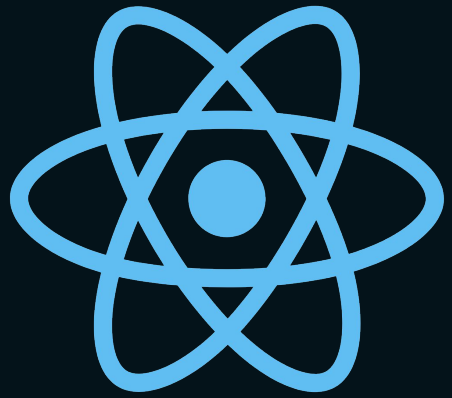
- Converta o componente de três propriedades em um componente de classe;
- Utilize a mesma estrutura, só mude a forma de criação do mesmo;



Ex 6: Componente com Estado

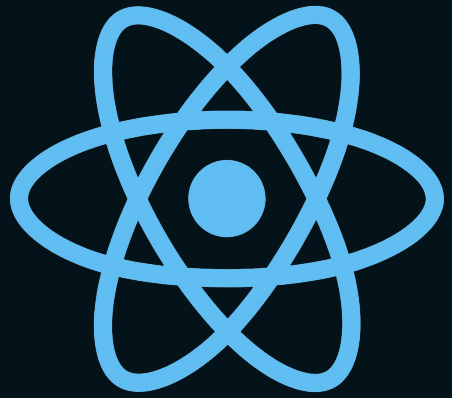
- Crie um componente chamado Counter;
- Ele deve renderizar um número e um botão;
- O número deve começar em 0, e incrementar em um a cada clique;
- Utilize o hook useState para gerenciar o estado do número;





Componentes e Props

Conclusão da seção

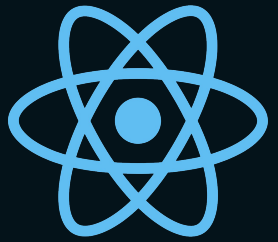


Estado e ciclo de vida

Introdução da seção

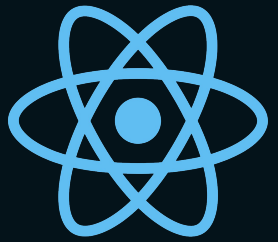
Ex 7: Usando estado

- Crie um componente chamado Toggle;
- Este componente deve ter um botão com um texto;
- O texto é alterado dinamicamente por meio de um estado;
- Deve começar com ON e ao clicar mudar para OFF;
- O contrário também deve ocorrer;



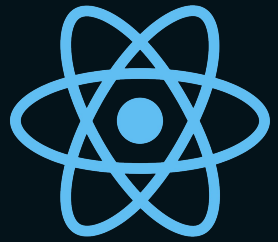
Ex 8: Múltiplos estados

- Crie um componente semelhante ao Toggle;
- Porém agora ele deve conter dois estados;
- O número de cliques no botão precisa ser exibido e atualizado a cada clique;



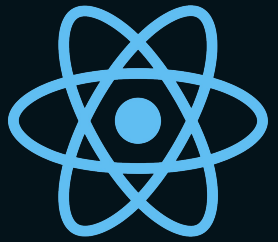
Ex 9: Relógio em tempo real

- Crie um componente Clock;
- Ele deve armazenar a hora atual em um state;
- Utilize o useEffect para atualizar a hora a cada um segundo;
- A hora pode ser obtida com a classe Date;



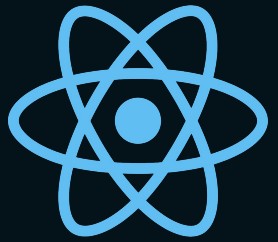
Ex 10: Lista de Tarefas

- Crie um componente chamado TodoList;
- Ele deve permitir que o usuário adicione uma tarefa a uma lista;
- Cada tarefa deve ter duas propriedades: id e task;
- Onde task é o título;
- Utilize o useState para gerenciar o estado da lista de tarefas;



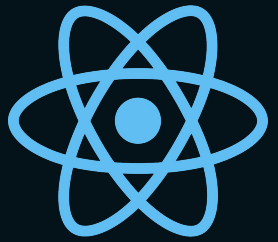
Ex 11: Filtro

- Implemente no componente do exercício anterior um filtro de tarefas;
- Um campo de texto que mostra apenas as tarefas que contenham o texto digitado;
- Utilize o useState para gerenciar o filtro;



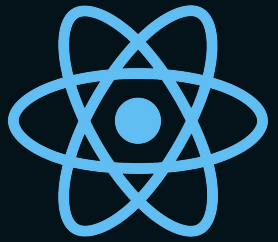
Ex 12: Verificador de largura da tela

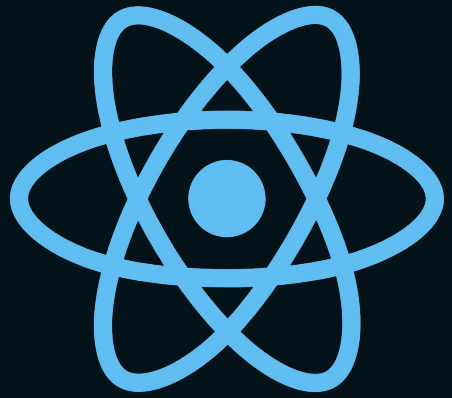
- Crie um novo componente que exibe a largura da tela;
- A cada mudança de largura a dimensão deve ser atualizada de forma instantânea;
- Utilize useState e useEffect para chegar no resultado final;



Ex 13: Simulação de busca de dados

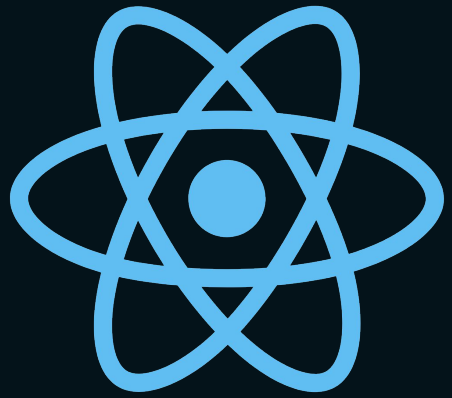
- Crie um componente que vai simular uma requisição a uma API;
- Utilize o setTimeout para representar a demora da resposta;
- Enquanto a resposta não chega, exiba um estado de loading;
- E depois exiba os dados;





Estado e ciclo de vida

Conclusão da seção

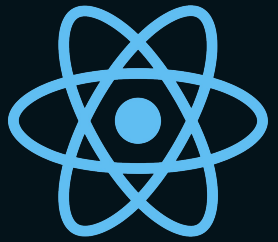


Eventos e Formulários

Introdução da seção

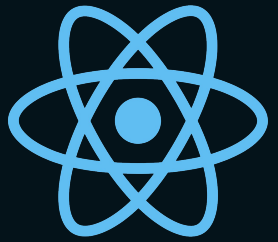
Ex 14: Formulário de Registro

- Crie um componente com um formulário de registro;
- Os campos são: nome, email e senha;
- Faça o envio do formulário;
- Em vez de enviar para um servidor, exiba todos os dados em um console.log;



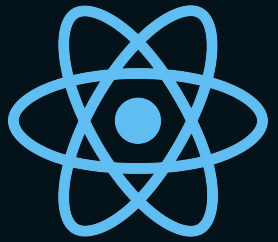
Ex 15: Formulário Dinâmico

- Crie um componente com um formulário;
- Adicione um botão que inclui um novo campo de input a cada clique nele;
- Faça o envio do formulário;
- Em vez de enviar para um servidor, exiba todos os dados em um console.log (inclusive dos inputs adicionados dinamicamente);



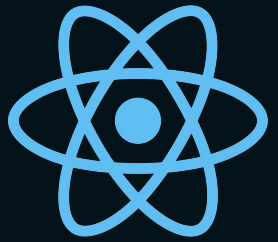
Ex 16: Validação de formulário

- Para este exercício utilize o formulário do Exercício 14 como base;
- Crie validações para todos os campos;
- A validação deve ser ativada quando o usuário digita no campo e quando envia o formulário;
- A mensagem do erro deve ser exibida embaixo de cada um dos inputs;



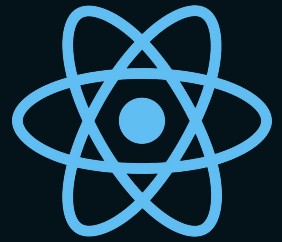
Ex 17: Upload de arquivo

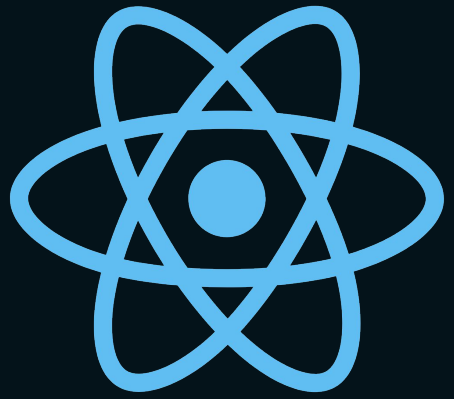
- Crie um componente que simula o upload de um arquivo;
- Exiba o nome do arquivo, após o upload no input do tipo file;
- E se for uma imagem, mostre o preview da mesma;



Ex 18: Formulário múltiplas etapas

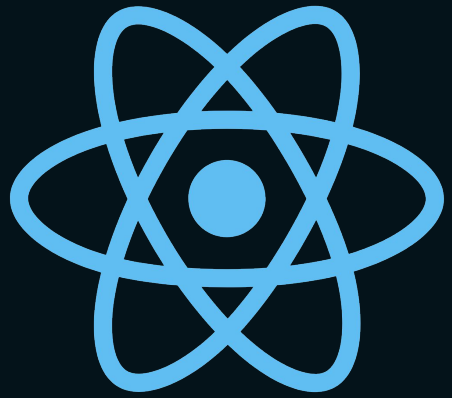
- Crie um componente de formulário que possui múltiplas etapas;
- Cada etapa deve ter um ou mais campos;
- Crie a funcionalidade de prosseguir e voltar nas etapas;
- Adicione uma validação mínima para os inputs em cada etapa;





Eventos e Formulários

Conclusão da seção

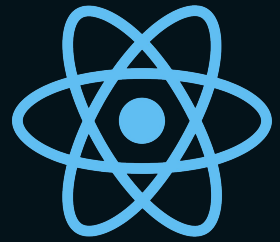


Estilização com CSS no React

Introdução da seção

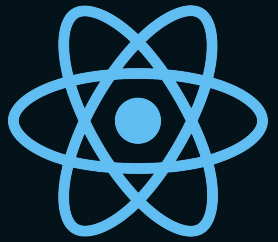
Ex 19: HelloWorld com estilos

- Crie um componente que exibe a mensagem Hello World;
- Utilize CSS inline para estilizar ele;
- Altere: cor, tamanho de fonte e cor de fundo do elemento;
- Armazene os estilos em uma variável e depois aplique no elemento;



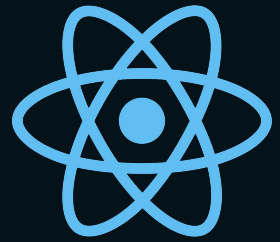
Ex 20: Estilo com CSS externo

- Crie um componente que tem um botão;
- Este botão deve ser estilizado com arquivo de CSS externo;
- Você deve criar um novo arquivo de CSS, não utilize os já iniciados pelo React;



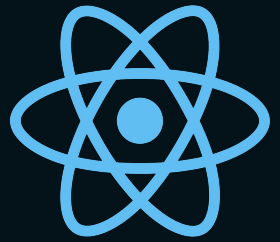
Ex 21: Tema light e dark

- Crie um componente que tem um botão;
- Este botão deve mudar o tema do projeto para light ou dark, quando o usuário clicar;
- Utilize CSS modules para implementar este exercício;



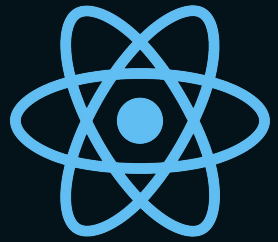
Ex 22: Styled Components

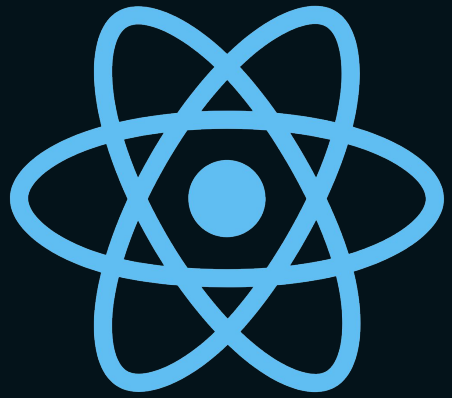
- Crie um novo componente;
- Faça estilos, com no mínimo 5 propriedades, utilizando styled components;
- Lembre-se de instalar a lib: styled-components;



Ex 23: SASS com React

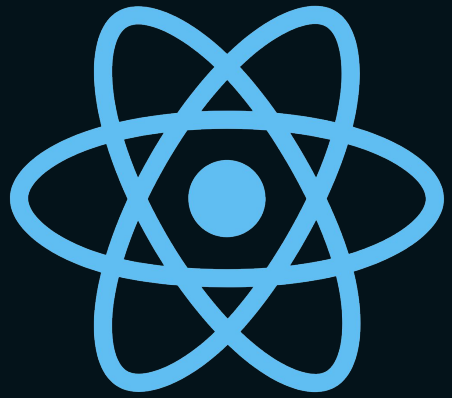
- Crie um novo componente;
- Utilize SASS para criar as regras de estilo dele;
- Aplique recursos como variáveis e nesting;
- Se você desejar implemente outros também;





Estilização com CSS no React

Conclusão da seção

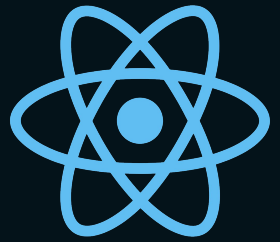


Roteamento e navegação

Introdução da seção

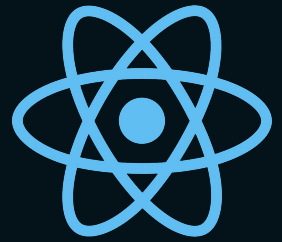
Ex 24: Navegação básica

- Para os exercícios desta seção utilize a lib “react-router-dom”;
- Crie três páginas;
- Faça a configuração de roteamento para elas;



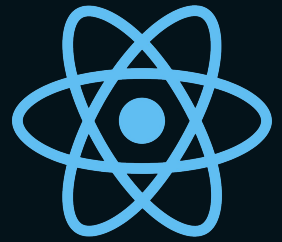
Ex 25: Rota parametrizada

- Crie uma nova rota;
- Esta deve aceitar um parâmetro pela URL;
- Exiba o valor do parâmetro no componente;



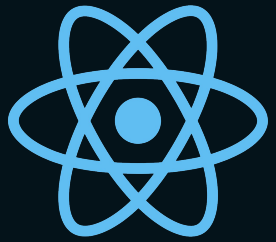
Ex 26: Links de navegação

- Crie um componente;
- Ele deve conter uma lista de links de navegação;
- Crie links para pelo menos os três primeiros componentes;
- Exiba este componente em todas as páginas;



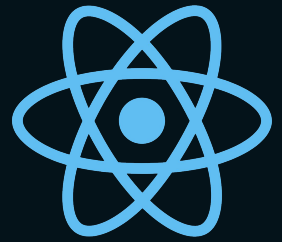
Ex 27: Rota de Erro 404

- Crie uma nova página;
- Ela servirá como uma rota 404;
- Através do React Router, redirecione todos os acessos para URLS que não existam a este componente;



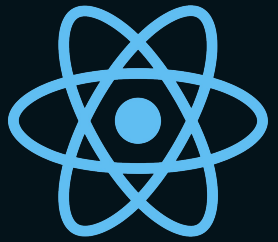
Ex 28: Rota com query params

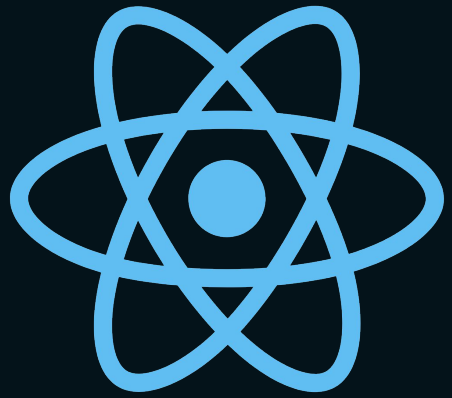
- Crie uma nova página que pode consultar os query params (parâmetros de URL);
- Ela deve exibir o resultado dos parâmetros;
- Crie um componente de busca, com um input;
- O conteúdo digitado deve ser enviado para a URL com query params;
- Este exercício simula um formulário de busca;



Ex 29: Rota protegida

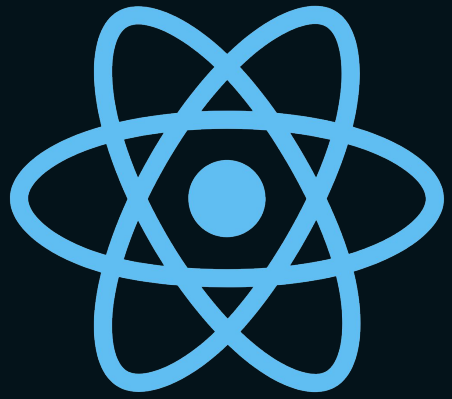
- Crie uma funcionalidade básica de autenticação;
- Onde uma das rotas deve ser protegida;
- Caso o usuário não esteja autenticado, você deve redirecionar ele para home;
- Para simplificar: controle a autenticação com uma variável do tipo boolean, true (autenticado) e false (não autenticado);





Roteamento e navegação

Conclusão da seção

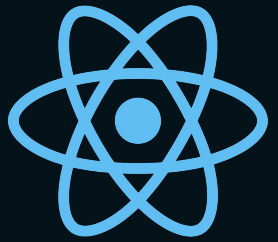


Condicionais e listas

Introdução da seção

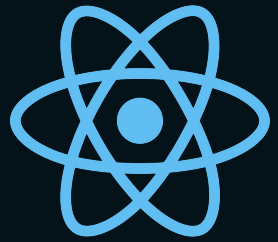
Ex 30: Renderização com ternário

- Crie um componente que contém uma condição ternária;
- Ele recebe uma prop com true ou false;
- Se for true deve renderizar um elemento, se for false outro;



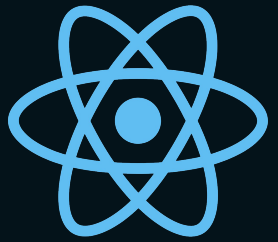
Ex 31: Renderização com switch

- Crie um componente que contém um switch;
- Ele deve receber uma prop que coordena o que o switch vai renderizar;
- Crie mais de dois cases, para renderizar elementos diferentes;



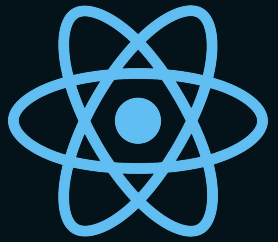
Ex 32: Listas aninhadas

- Crie um componente que renderiza uma lista;
- Esta lista é baseada num array que contém outras listas, ou seja, array de arrays;
- A estrutura deve ficar um loop de lista, com outro loop interno de listas;



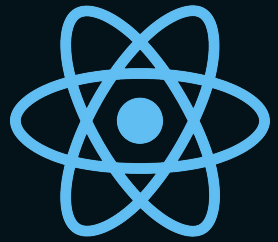
Ex 33: Componente baseado em tempo

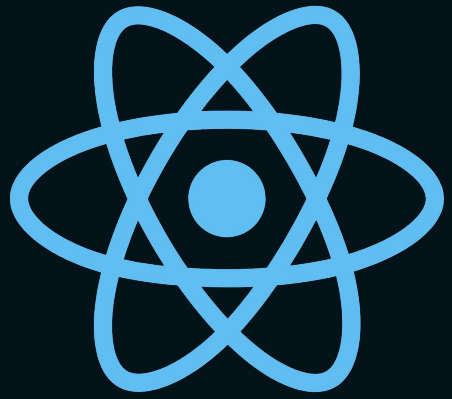
- Crie um novo componente;
- Ele deve consultar a hora do sistema, pode-se utilizar a classe Date;
- Baseado no período: manhã, tarde ou noite, Exiba: “Bom dia”, “Boa tarde” ou “Boa noite”;



Ex 34: Multi seleção

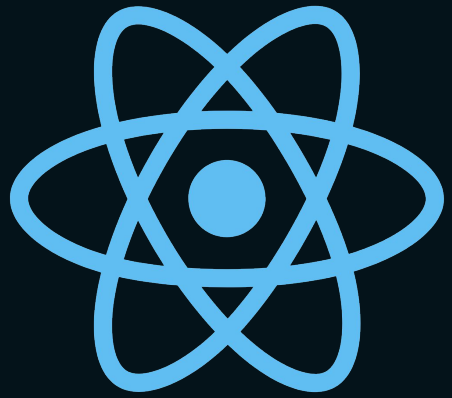
- Crie um componente que renderiza uma lista;
- Ele deve ter três funcionalidades;
- 1 - Marcar um item selecionado por clique;
- 2 - Marcar todos os itens;
- 3 - Desmarcar todos os itens;
- Você pode colocar um estilo inline ou classe para marcar os elementos selecionados;





Condicionais e listas

Conclusão da seção

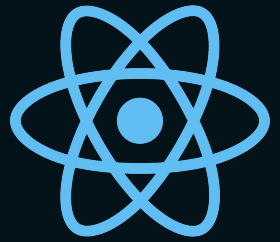


Gerenciamento de contexto

Introdução da seção

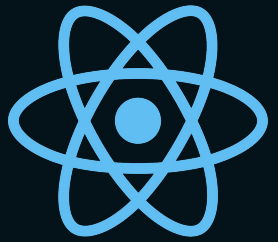
Ex 35: Contador com Redux

- Crie um componente de contador;
- O estado dele deve ser gerenciador por Redux;
- Você precisa criar as funcionalidades de: incrementar, decrementar e resetar o contador;



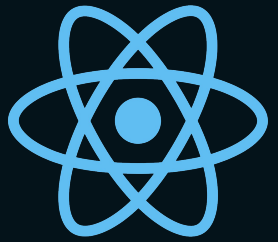
Ex 36: Todo com Context API

- Crie uma Lista de Tarefas gerenciada pelo Context API;
- Deve ser possível criar tarefas, remover e também marcar como completa;
- O formulário e a lista devem estar em componentes separados;
- As funcionalidades devem estar no arquivo do contexto;



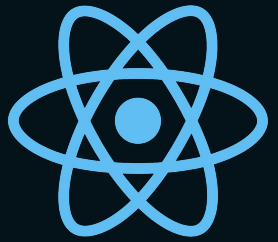
Ex 37: Paginação com Redux

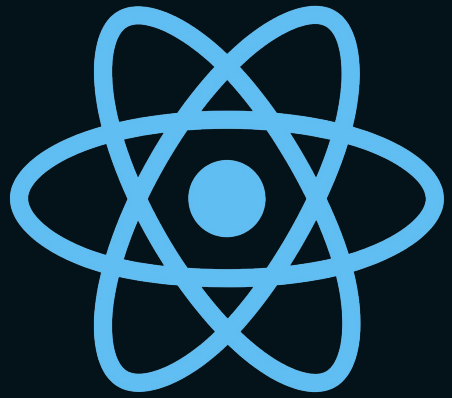
- Crie um componente com paginação, que deve ser gerenciado por Redux;
- Utilize uma API como JSONPlaceholder ou dados mockados, para simular a mudança de páginas;
- A chamada de API deve ser controlada pelo Redux;
- Para facilitar o manuseio da requisição, você pode utilizar o pacote redux tookit;
- Crie a estrutura com: actions, reducers e store;



Ex 38: Formulário com React Final Form

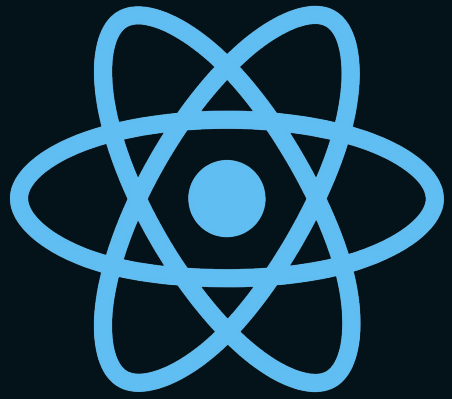
- Crie um formulário com auxílio do pacote React Final Form;
- Ele deve conter os campos: nome, sobrenome, idade;
- Se a idade for menor que 18, exiba um campo para adicionar o nome do responsável;
- Faça validação em todos os campos;
- Crie um botão para envio e outro para resetar os dados;
- Ao enviar, exiba os dados no console;





Gerenciamento de contexto

Conclusão da seção

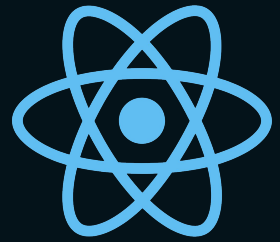


Testes e depuração

Introdução da seção

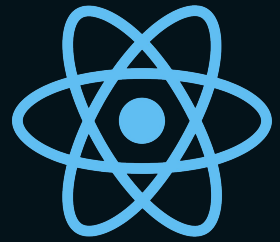
Ex 39: Testes em componente

- Crie um componente;
- Crie um teste simples para o mesmo;
- Utilize React Testing Library;



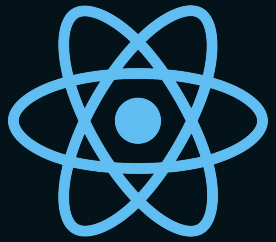
Ex 40: Testes simulação de evento

- Crie um componente com um botão;
- Ele deve mudar o texto quando clicado;
- O teste deve verificar se o texto foi alterado;



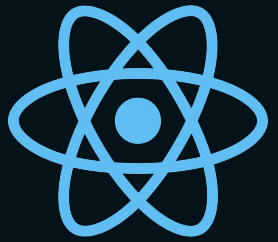
Ex 41: Testes de formulário

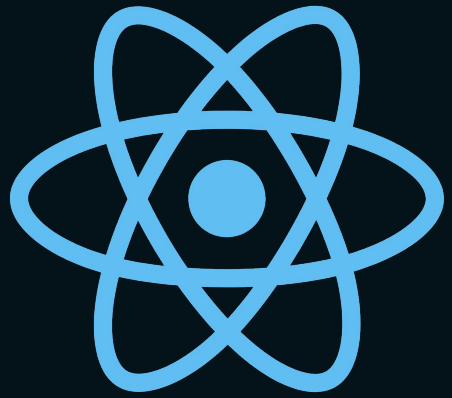
- Crie um formulário com um botão de envio;
- Crie um teste que verifica se a submissão do formulário funciona corretamente;
- O teste deve preencher o input com um texto, e verificar se o texto está presente;



Ex 42: Teste com Mock de API

- Escreva um teste que simula uma chamada de API;
- Você pode usar o pacote: jest-fetch-mock;
- O componente deve conter um estado de loading, e o teste deve evidenciar se ele foi removido após o carregamento de dados;
- Além disso, preencha com dados da API algum elemento e verifique se isso ocorreu com o teste;





Testes e depuração

Conclusão da seção