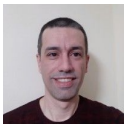


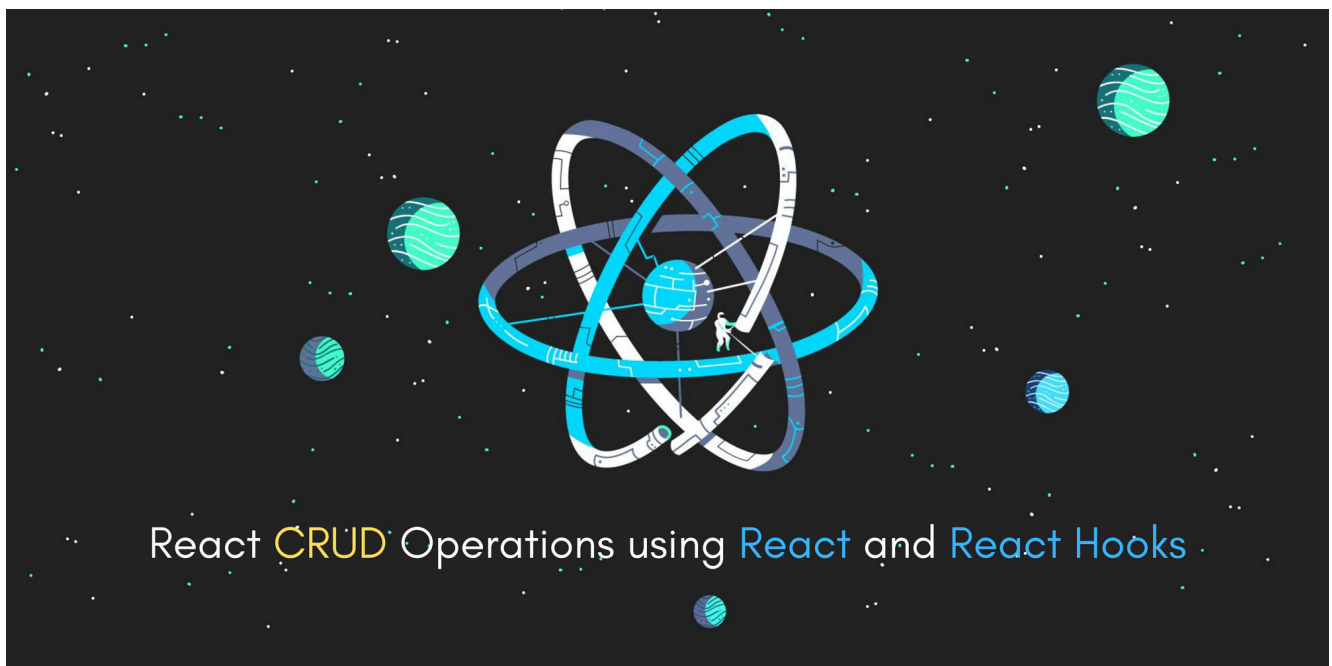
Aprenda a programar — currículo gratuito de 3 mil horas

29 DE MARÇO DE 2022 / #REACT

# Como realizar operações de CRUD usando React, hooks do React e Axios



Daniel Rosa



**Artigo original:**

<https://www.freecodecamp.org/news/how-to-perform-crud-operations-using-react/>

Se você está trabalhando com React, pode ser difícil entender e implementar solicitações de API.

Aprenda a programar — currículo gratuito de 3 mil horas

Vamos lá.

## Como instalar o Node e o npm

Primeiramente, vamos instalar o Node em nosso sistema. Vamos usá-lo, primeiramente, para executar nosso código em JavaScript.

Para baixar o Node, acesse <https://nodejs.org/en/>.

Você também precisará do **node package manager**, ou npm, que já vem integrado ao Node. Você pode usá-lo para instalar os pacotes para seus apps em JavaScript. Felizmente, como ele vem com o Node, não é necessário baixá-lo separadamente.

Depois de baixá-los e instalá-los, abra seu terminal ou o prompt de comando e digite `node -v`. Assim, você poderá conferir a versão do Node que você tem.

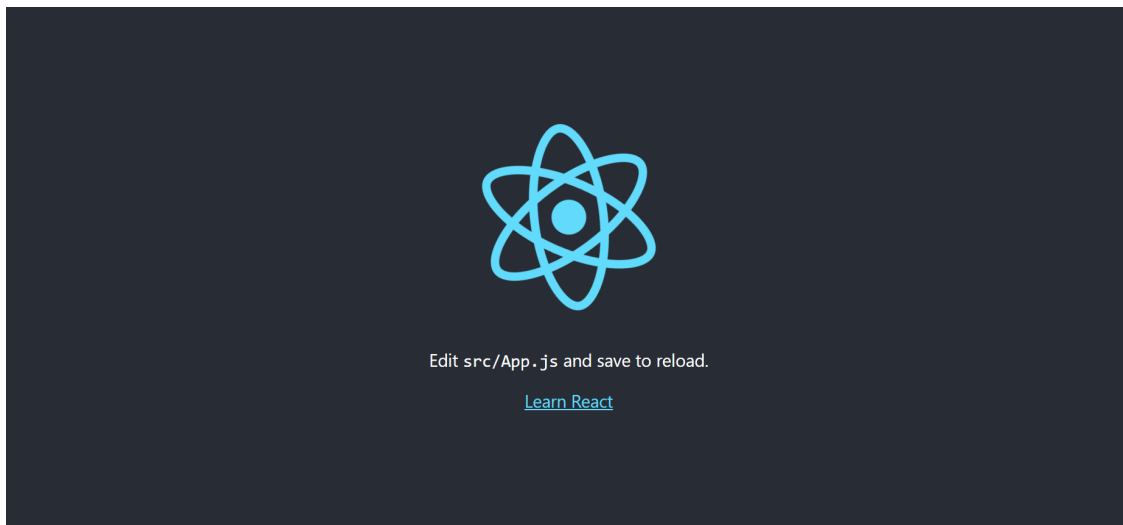
## Como criar sua aplicação em React

Para criar sua aplicação em React, digite `npx-create-react-app <nome-do-seu-app>` no seu terminal ou `npx-create-react-app react-crud`, neste caso.

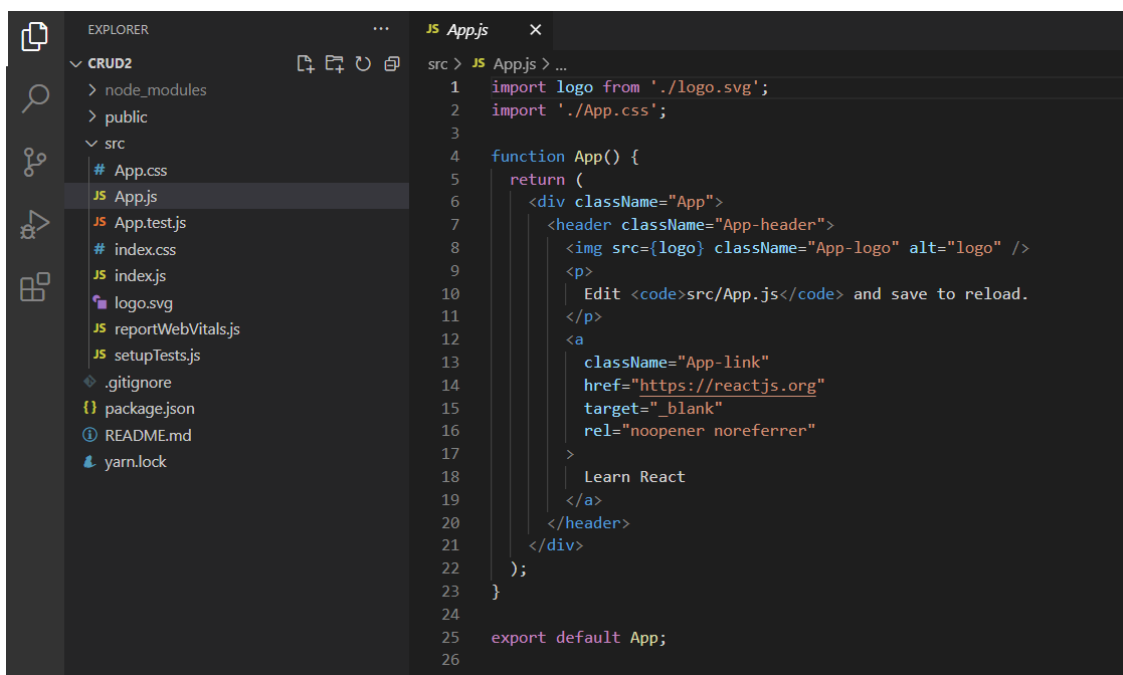
Você verá os pacotes serem instalados.

Ao terminar de baixar os pacotes, vá até a pasta do projeto e digite `npm start`.

Aprenda a programar — currículo gratuito de 3 mil horas



### O boilerplate padrão do React



### Nosso arquivo App.js

## Como instalar o pacote Semantic UI para o React

Vamos instalar o pacote Semantic UI para o React em nosso projeto. Semantic UI é uma biblioteca de UI (interface de usuário) feita para

Aprenda a programar — currículo gratuito de 3 mil horas

Voce pode instalar o pacote usando um dos comandos abaixo, dependendo do seu gerenciador de pacotes.

```
yarn add semantic-ui-react semantic-ui-css
```

Para o gerenciador de pacotes do Yarn

```
npm install semantic-ui-react semantic-ui-css
```

Para o gerenciador de pacotes do Node, o NPM

Além disso, importe a biblioteca em seu arquivo de entrada principal do app, chamado index.js.

```
import 'semantic-ui-css/semantic.min.css'
```

Cole isto no seu arquivo index.js

## Como criar sua aplicação com CRUD

Agora, vamos começar a criar nossa aplicação com CRUD usando o React.

## Aprenda a programar — currículo gratuito de 3 mil horas

```
import './App.css';

function App() {
  return (
    <div>
      React Crud Operations
    </div>
  );
}

export default App;
```

Adicionando um título à nossa aplicação

Agora, vamos garantir que ela esteja centralizada.

Dê à div pai a classname main. No arquivo App.css, usaremos Flexbox para centralizar o título.

```
import './App.css';

function App() {
  return (
    <div className="main">
      React Crud Operations
    </div>
  );
}

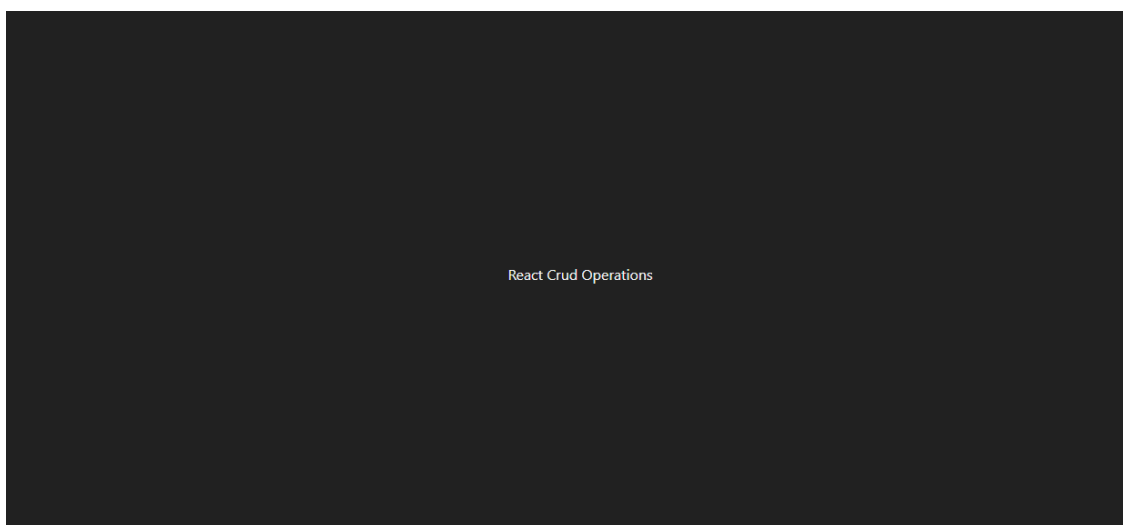
export default App;
```

app.js com a className main na div pai

## Aprenda a programar — currículo gratuito de 3 mil horas

```
justify-content: center;  
align-items: center;  
height: 100vh;  
}
```

Nosso arquivo app.css



Nosso título, agora, está centralizado com perfeição.

Agora que a aparência está melhor, precisamos colocá-lo em realce e adicionar umas fontes legais. Para fazer isso, usaremos as tags ao redor do título, assim:

```
import './App.css';  
  
function App() {  
  return (  
    <div className="main">  
      <h2 className="main-header">React Crud Operations</h2>  
    </div>  
  );  
}
```

## Aprenda a programar — currículo gratuito de 3 mil horas

Vamos importar uma Google Font. Acesse <https://fonts.google.com/> para escolhermos uma.

Selecione a fonte de sua preferência. Para o exemplo, usaremos a família de fontes Montserrat.

Importe a sua fonte preferida no arquivo App.css, assim:

```
@import url('https://fonts.googleapis.com/css2?family=Montserrat');
```

Agora, vamos mudar a fonte do título.

```
<div className="main">
  <h2 className="main-header">React Crud Operations</h2>
</div>
```

Dê à tag do título a `className main-header`, assim.

Em seguida, no seu App.css, adicione a família da fonte:

```
.main-header{
  font-family: 'Montserrat', sans-serif;
}
```

Aprenda a programar — currículo gratuito de 3 mil horas

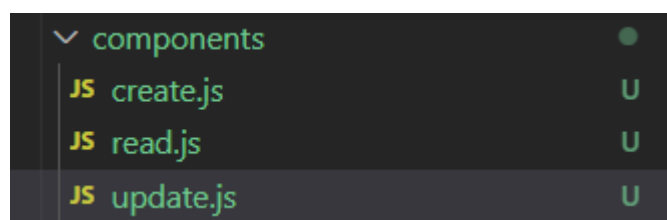
## React Crud Operations

Agora, você verá o título modificado.

# Como criar seus componentes de CRUD

Vamos criar quatro componentes de CRUD – Create, Read, Update e Delete (criar, ler, atualizar e excluir, respectivamente).

Na nossa pasta *src*, criamos uma pasta chamada *components*. Dentro dela, criamos três arquivos – *create.js*, *read.js* e *update.js*. Para a exclusão (*delete*), não precisamos de um componente adicional.



Agora, vamos implementar a operação de criação.

Para isso, precisamos usar APIs *mock* ou de simulação. Essas APIs enviarão dados ao servidor falso que criaremos, apenas para fins didáticos.



## Aprenda a programar — currículo gratuito de 3 mil horas

API

**mockapi.io**The easiest way to mock REST APIs! (check out [docs](#))

GET STARTED

DEMO PROJECT

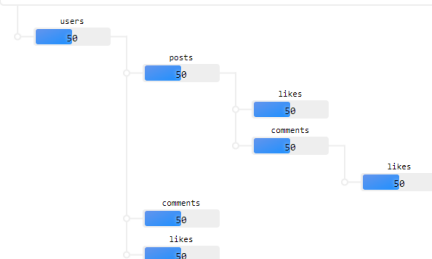
API endpoint

<https://SECRET.mockapi.io/:endpoint>

NEW RESOURCE

GENERATE ALL

RESET ALL



MockAPI

Criar um projeto clicando no botão +.



Clique no botão + para criar um novo projeto

## Aprenda a programar — currículo gratuito de 3 mil horas

**Project name**

Example: Todo App, Project X...

**API Prefix (optional)**

Add API prefix to all endpoints in this project.

Example: /api/v1

**Collaborators (optional)**

Collaborators can `create`, `update`, and `delete` resources in this project.

Search by name...

CREATE

CANCEL

Adicione o nome do projeto e clique no botão Create (Criar).

Projects → 

C

 CRUD

**API endpoint**

https://60fbca4591156a0017b4c8a7.mockapi.io/:endpoint

NEW RESOURCE

GENERATE ALL

RESET ALL

🔍 No resources yet...

Em seguida, crie um novo recurso clicando no botão NEW RESOURCE (Novo recurso).

## Aprenda a programar — currículo gratuito de 3 mil horas

ndpoint  
//60fbca459

RESOURCE

Resource name

Enter meaningful resource name, it will be used to generate API endpoints.

Example: users, comments, articles...

Schema (optional)

Define Resource schema, it will be used to generate mock data.

id

Object ID

createdAt

Faker.js

Recent

name

Faker.js

Find name

avatar

Faker.js

Avatar

+

Object template (optional)

To define more complex structure for your data use JSON template. You can reference Faker.js methods using `\$.`.

```
{
  "username": "$internet.userName",
  "knownIps": ["$internet.ip", "$internet.ipv6"],
  "profile": {
    "firstName": "$name.firstName",
    "lastName": "$name.lastName",
    "staticData": [100, 200, 300]
  }
}
```

EXAMPLE

JSON template

CREATE

CANCEL

Será solicitado o nome do recurso (em *Resource Name*). Coloque ali o nome que achar apropriado.

<https://www.freecodecamp.org/portuguese/news/como-realizar-operacoes-de-crud-usando-react-hooks-do-react-e-axios/>

11/48

## Aprenda a programar — currículo gratuito de 3 mil horas

/60fbca459

Enter meaningful resource name, it will be used to generate API endpoints.

fakeData

Schema (optional)

Define Resource schema, it will be used to generate mock data.

id Object ID

+

Object template (optional)

To define more complex structure for your data use JSON template. You can reference Faker.js methods using `\$.`.

```
{
  "username": "$internet.userName",
  "knownIps": ["$internet.ip", "$internet.ipv6"],
  "profile": {
    "firstName": "$name.firstName",
    "lastName": "$name.lastName",
    "staticData": [100, 200, 300]
  }
}
```

EXAMPLE

JSON template

Endpoints

Enable/disable endpoints and customize response JSON.

By default Mockapi will return either a list of items or a single item depending on

CREATE CANCEL

Remova os campos adicionais, como name, avatar e createdAt, pois não precisaremos deles. Então, clique em Create (Criar).

mock API Docs

Projects → C CRUD

API endpoint  
https://60fbca4591156a0017b4c8a7.mockapi.io/:endpoint

NEW RESOURCE GENERATE ALL RESET ALL

fakeData  
0 Data Edit Delete

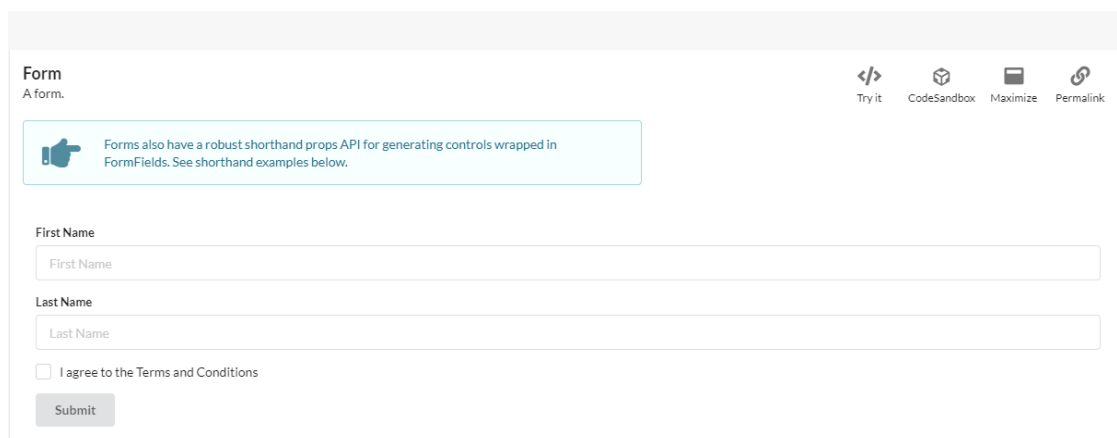
Aprenda a programar — currículo gratuito de 3 mil horas

Clique em fakeData e você verá a API sendo aberta em uma nova guia. O banco de dados agora está vazio.

## Como criar um formulário para o componente Create

Vamos usar um formulário da biblioteca Semantic UI.

Vá até Semantic UI, e procure por *Form* na barra de pesquisa à esquerda.



The screenshot shows the Semantic UI documentation page for the 'Form' component. The page title is 'Form' with the subtitle 'A form.'. In the top right corner, there are links: 'Try it', 'CodeSandbox', 'Maximize', and 'Permalink'. A light blue callout box contains a thumbs-up icon and the text: 'Forms also have a robust shorthand props API for generating controls wrapped in FormFields. See shorthand examples below.' The form itself consists of two text input fields labeled 'First Name' and 'Last Name'. Below these fields is a checkbox labeled 'I agree to the Terms and Conditions'. At the bottom of the form is a 'Submit' button.

Você verá um formulário assim. Clique em Try it (Experimente) na parte superior esquerda para obter o código.

## Aprenda a programar — currículo gratuito de 3 mil horas

First Name

Last Name

☐ I agree to the Terms and Conditions

Submit

```
1 import React from 'react'
2 import { Button, Checkbox, Form } from 'semantic-ui-react'
3
4 const FormExampleForm = () => (
5   <Form>
6     <Form.Field>
7       <label>First Name</label>
8       <input placeholder='First Name' />
9     </Form.Field>
10    <Form.Field>
11      <label>Last Name</label>
12      <input placeholder='Last Name' />
13    </Form.Field>
14    <Form.Field>
15      <Checkbox label='I agree to the Terms and Conditions' />
16    </Form.Field>
17    <Button type='submit'>Submit</Button>
18  </Form>
19 )
20
21 export default FormExampleForm
22
```

Copie este código e cole-o no seu arquivo Create.js, assim:

```
import React from 'react'
import { Button, Checkbox, Form } from 'semantic-ui-react'

const Create = () => (
  <Form>
    <Form.Field>
      <label>First Name</label>
      <input placeholder='First Name' />
    </Form.Field>
    <Form.Field>
      <label>Last Name</label>
      <input placeholder='Last Name' />
    </Form.Field>
    <Form.Field>
      <Checkbox label='I agree to the Terms and Conditions' />
    </Form.Field>
    <Button type='submit'>Submit</Button>
  </Form>
)

export default Create;
```



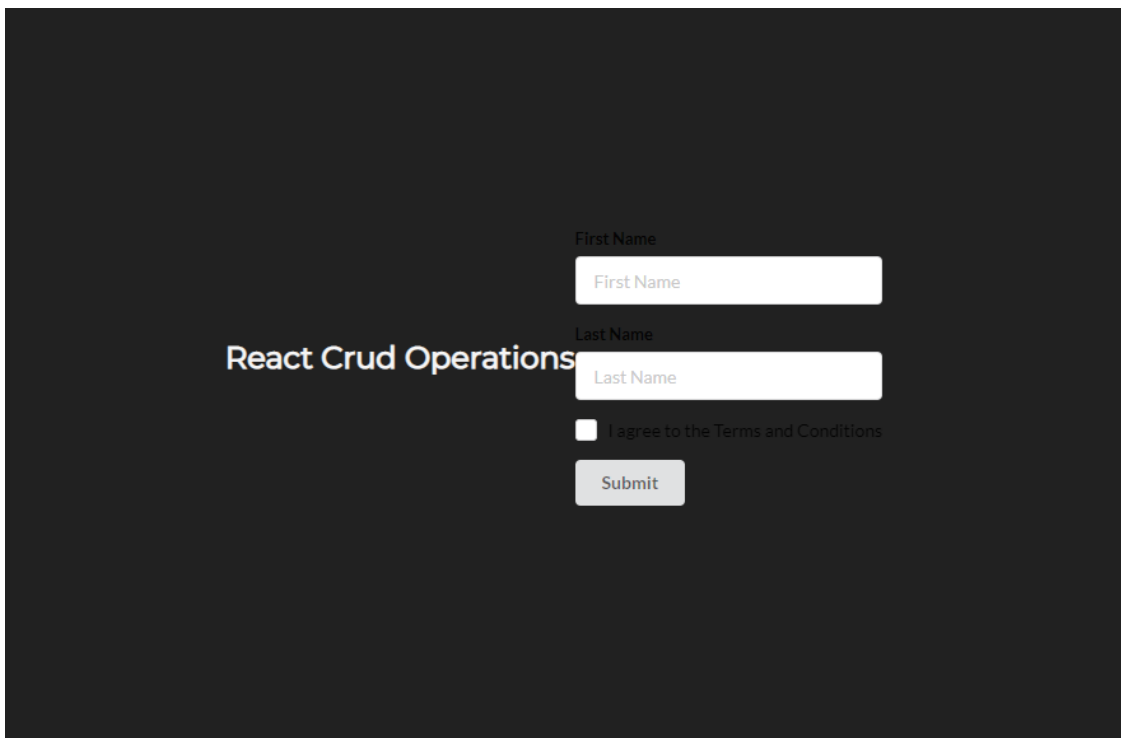
## Aprenda a programar — currículo gratuito de 3 mil horas

```
import './App.css';
import Create from './components/create';

function App() {
  return (
    <div className="main">
      <h2 className="main-header">React Crud Operations</h2>
      <div>
        <Create/>
      </div>
    </div>
  );
}

export default App;
```

Este é o resultado:



The screenshot shows a web application titled "React Crud Operations" on a dark background. The title is displayed in a large, white, serif font. To the right of the title is a form with two input fields: "First Name" and "Last Name", both with white borders and placeholder text. Below these fields is a checkbox labeled "I agree to the Terms and Conditions". At the bottom of the form is a "Submit" button with a white border and text.

## Aprenda a programar — currículo gratuito de 3 mil horas

No arquivo `Create.js`, dê à **Form** a `className` de `create-form`.

```
import React from 'react'
import { Button, Checkbox, Form } from 'semantic-ui-react'

const Create = () => (
  <Form className="create-form">
    <Form.Field>
      <label>First Name</label>
      <input placeholder='First Name' />
    </Form.Field>
    <Form.Field>
      <label>Last Name</label>
      <input placeholder='Last Name' />
    </Form.Field>
    <Form.Field>
      <Checkbox label='I agree to the Terms and Conditions' />
    </Form.Field>
    <Button type='submit'>Submit</Button>
  </Form>
)

export default Create;
```

app.js

E adicione a classe abaixo ao seu arquivo `App.css`:

```
.create-form label{
  color: whitesmoke !important;
  font-family: 'Montserrat', sans-serif;
  font-size: 12px !important;
}
```



## Aprenda a programar — currículo gratuito de 3 mil horas

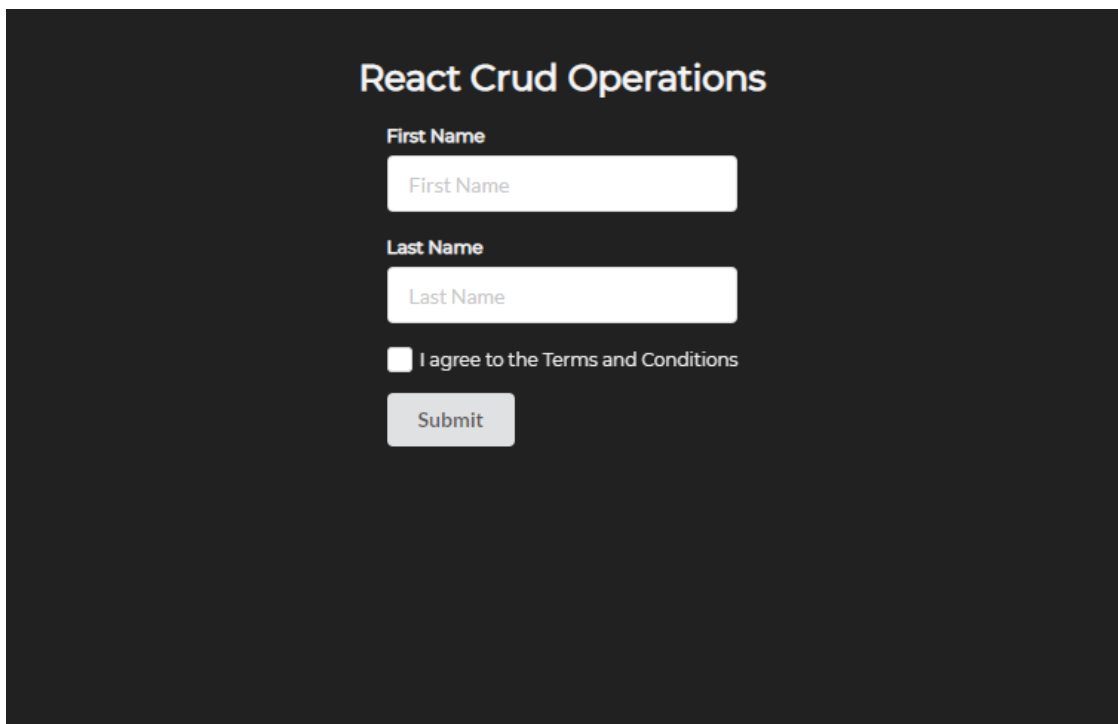
Essa classe terá como filhos todos os labels dos campos de formulários e aplicará a eles a cor whitesmoke. Ela também mudará a fonte e aumentará o tamanho da fonte.

Em nosso `className main`, adicione a propriedade `flex-direction`. Essa propriedade definirá a orientação como `column`, de modo que cada elemento na `className main` seja alinhado na horizontal.

```
.main{
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #212121;
  color: whitesmoke;
  flex-direction: column;
}
```

App.css

Aprenda a programar — currículo gratuito de 3 mil horas



The screenshot shows a web form titled "React Crud Operations" on a dark background. The form contains the following elements:

- A label "First Name" above a text input field with the placeholder text "First Name".
- A label "Last Name" above a text input field with the placeholder text "Last Name".
- A checkbox followed by the text "I agree to the Terms and Conditions".
- A "Submit" button.

O formulário, agora, tem uma aparência bem melhor.

Em seguida, vamos obter os dados dos campos do formulário e colocá-los em nosso console. Para isso, usaremos o hook `useState` em React.

No arquivo `Create.js`, importe `useState` de React.

```
import React, { useState } from 'react';
```

Depois, crie states para *first name*, *last name* (nome e sobrenome, respectivamente) e para a caixa de seleção. Inicializaremos os states como vazios ou *false*.

## Aprenda a programar — currículo gratuito de 3 mil horas

```
const [firstName, setFirstName] = useState('');
const [lastName, setLastName] = useState('');
const [checkbox, setCheckbox] = useState(false);
return (
  <div>
    <Form className="create-form">
      <Form.Field>
        <label>First Name</label>
        <input placeholder='First Name' />
      </Form.Field>
      <Form.Field>
        <label>Last Name</label>
        <input placeholder='Last Name' />
      </Form.Field>
      <Form.Field>
        <Checkbox label='I agree to the Terms and Cor
      </Form.Field>
      <Button type='submit'>Submit</Button>
    </Form>
  </div>
)
```

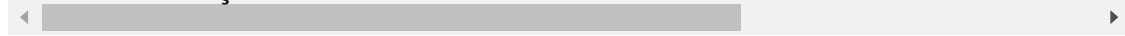
Você poderá ver agora que o código está agindo como um componente funcional. Por isso, precisamos transformar o componente em um componente funcional. Somente poderemos usar hooks com esse tipo de componente.

Vamos configurar *first name*, *last name* e a caixa de seleção usando as propriedades `setFirstName`, `setLastName` e `setCheckbox`, respectivamente.

```
<input placeholder='First Name' onChange={(e) => setFirstName(e.tai
<input placeholder='Last Name' onChange={(e) => setLastName(e.tai
```

## Aprenda a programar — currículo gratuito de 3 mil horas

Agora, estamos capturando os *states* de *first name*, *last name* e da caixa de seleção.



Crie uma função chamada `postData`, que usaremos para enviar dados para a API. Dentro da função, escreveremos este código:

```
const postData = () => {  
  console.log(firstName);  
  console.log(lastName);  
  console.log(checkbox);  
}
```

Estamos imprimindo no console o conteúdo de *first name*, *last name* e da caixa de seleção.

No botão Submit, atribua essa função usando o evento `onClick` para que, quando o botão Submit for pressionado, essa função seja chamada.

```
<Button onClick={postData} type='submit'>Submit</Button>
```

Aqui temos o código completo para o arquivo *create.js*:

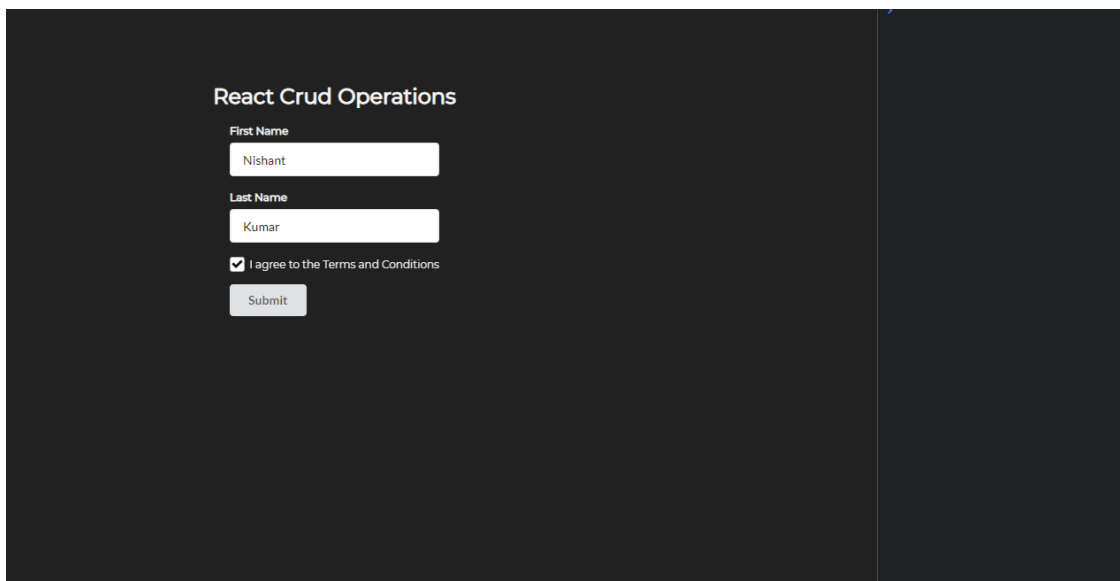
```
import React, { useState } from 'react';  
import { Button, Checkbox, Form } from 'semantic-ui-react'  
  
export default function Create() {  
  const [firstName, setFirstName] = useState('');
```

## Aprenda a programar — currículo gratuito de 3 mil horas

```
        console.log(checkbox);
    }
    return (
      <div>
        <Form className="create-form">
          <Form.Field>
            <label>First Name</label>
            <input placeholder='First Name' onChange={(e)
          </Form.Field>
          <Form.Field>
            <label>Last Name</label>
            <input placeholder='Last Name' onChange={(e)
          </Form.Field>
          <Form.Field>
            <Checkbox label='I agree to the Terms and Cor
          </Form.Field>
          <Button onClick={postData} type='submit'>Submit<,
        </Form>
      </div>
    )
  }
}
```

Digite algum valor em *first name* e *last name* e marque a caixa de seleção. Em seguida, clique no botão Submit. Você verá os dados aparecerem no console, assim:

Aprenda a programar — currículo gratuito de 3 mil horas

A screenshot of a web application titled "React Crud Operations". It features a dark background with a light-colored form. The form has two input fields: "First Name" with the value "Nishant" and "Last Name" with the value "Kumar". Below these is a checkbox labeled "I agree to the Terms and Conditions" which is checked. At the bottom of the form is a "Submit" button.

## Como usar o Axios para enviar solicitações de API para as Mock APIs

Vamos usar o Axios para enviar os dados do formulário para o servidor mock.

Primeiro, porém, precisamos instalá-lo.

Apenas digite `npm i axios` para instalar o pacote.

## Aprenda a programar — currículo gratuito de 3 mil horas

```
32
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\crud2> npm i axios
[.....] \ idealTree: timing arborist:ctor Completed in 7ms
```

Após termos instalado o pacote, vamos fazer a operação de criação.

Importe o Axios na parte superior do arquivo.

```
import axios from 'axios';
```

Importação do Axios

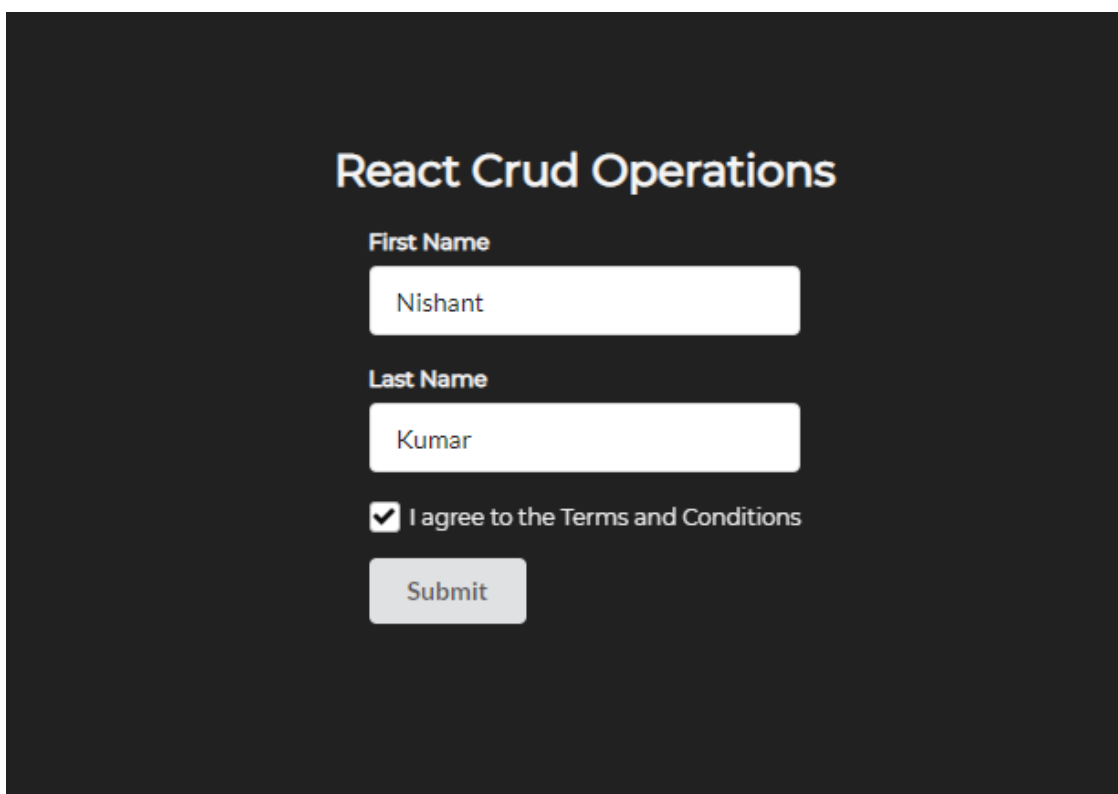
Na função `postData`, usaremos o Axios para enviar a solicitação de POST.

```
const postData = () => {
  axios.post(`https://60fbca4591156a0017b4c8a7.mockapi.io/1
    firstName,
    lastName,
    checkbox
  })
}
```

Enviando a solicitação de Post

Aprenda a programar — currículo gratuito de 3 mil horas

Ao clicarmos em Submit, essa função será chamada e enviará os dados ao servidor da API.



React Crud Operations

First Name

Nishant

Last Name

Kumar

☒ I agree to the Terms and Conditions

Submit

Insira seu nome, sobrenome e marque a caixa de seleção. Clique em Submit.

```
[{"id": "1", "firstName": "Nishant", "lastName": "Kumar", "checkbox": true}]
```

Se você verificar a API, verá seu nome, sobrenome e a caixa de seleção (ou *checkbox*, em inglês) assinalada como *true*, dentro do objeto.



### Aprenda a programar — currículo gratuito de 3 mil horas

Para iniciar a operação de leitura (Read), precisamos criar uma página de leitura. Também precisaremos do pacote React Router para navegar entre páginas diferentes.

Acesse <https://reactrouter.com/web/guides/quick-start> e instale o pacote usando `npm i react-router-dom`.

Depois de ele ter sido instalado, importamos algumas coisas do React Router:

```
import { BrowserRouter as Router, Route } from 'react-router-dom'
```



Importando Browser Router como Router e Route do pacote React Router

Em nosso App.js, envolvemos todo o *return* em um Router (roteador). O que isso significa, basicamente, é que tudo o que estiver dentro desse Router poderá usar roteamento em React.

```
import './App.css';
import Create from './components/create';
import { BrowserRouter as Router, Route } from 'react-router-dom'

function App() {
  return (
    <Router>
      <div className="main">
        <h2 className="main-header">React Crud Operations</h2>
        <div>
          <Create />
        </div>
      </div>
    </Router>
  )
}
```

## Aprenda a programar — currículo gratuito de 3 mil horas

Nosso App.js terá, agora, a aparência que vemos acima.

Substitua o Create dentro do return e adicione o código a seguir:

```
import './App.css';
import Create from './components/create';
import { BrowserRouter as Router, Route } from 'react-router-dom';

function App() {
  return (
    <Router>
      <div className="main">
        <h2 className="main-header">React Crud Operations</h2>
        <div>
          <Route exact path="/create" component={Create} />
        </div>
      </div>
    </Router>
  );
}

export default App;
```

Aqui, estamos usando o componente Route como Create. Definimos o caminho de Create como '/create'. Assim, se acessarmos <http://localhost:3000/create>, veremos a página de criação.

Do mesmo modo, precisamos das rotas para leitura e atualização.

```
import './App.css';
import Create from './components/create';
```

## Aprenda a programar — currículo gratuito de 3 mil horas

```
    ...`
  return (
    <Router>
      <div className="main">
        <h2 className="main-header">React Crud Operations</h2>
        <div>
          <Route exact path="/create" component={Create} />
        </div>
        <div style={{ marginTop: 20 }}>
          <Route exact path="/read" component={Read} />
        </div>

        <Route path="/update" component={Update} />
      </div>
    </Router>
  );
}

export default App;
```

Assim, crie as rotas de leitura e atualização (*read* e *update*) da mesma forma que você vê acima.

Se você acessar <http://localhost:3000/read>, verá o seguinte:



Rota de leitura (Read)

Aprenda a programar — currículo gratuito de 3 mil horas



Rota de atualização (Update)

## A operação de leitura

Para a operação de leitura (Read), precisaremos de um componente Table (tabela). Então, vá para a Semantic UI do React e use uma tabela da biblioteca.

```
import React from 'react';
import { Table } from 'semantic-ui-react'
export default function Read() {
  return (
    <div>
      <Table singleLine>
        <Table.Header>
          <Table.Row>
            <Table.HeaderCell>Name</Table.HeaderCell>
            <Table.HeaderCell>Registration Date</Table.HeaderCell>
            <Table.HeaderCell>E-mail address</Table.HeaderCell>
            <Table.HeaderCell>Premium Plan</Table.HeaderCell>
          </Table.Row>
        </Table.Header>
      </Table>
    </div>
  )
}
```

## Aprenda a programar — currículo gratuito de 3 mil horas

```

        <Table.Cell>jhlilk22@yahoo.com</Table.Ce
        <Table.Cell>No</Table.Cell>
    </Table.Row>
    <Table.Row>
        <Table.Cell>Jamie Harington</Table.Cell>
        <Table.Cell>January 11, 2014</Table.Cell>
        <Table.Cell>jamieharingonton@yahoo.com</
        <Table.Cell>Yes</Table.Cell>
    </Table.Row>
    <Table.Row>
        <Table.Cell>Jill Lewis</Table.Cell>
        <Table.Cell>May 11, 2014</Table.Cell>
        <Table.Cell>jilsewris22@yahoo.com</Table
        <Table.Cell>Yes</Table.Cell>
    </Table.Row>
</Table.Body>
</Table>
</div>
    )
}

```

Read.js

Aqui, você pode ver uma tabela com alguns dados de teste. Temos apenas uma linha de tabela. Assim, removeremos o resto.

```

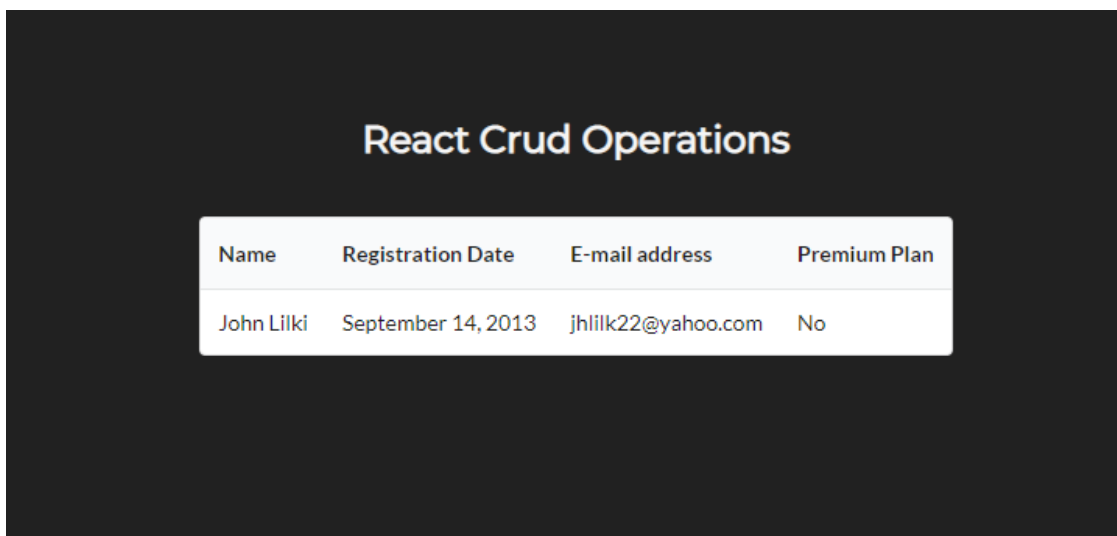
import React from 'react';
import { Table } from 'semantic-ui-react'
export default function Read() {
    return (
        <div>
            <Table singleLine>
                <Table.Header>
                    <Table.Row>
                        <Table.HeaderCell>Name</Table.HeaderCell>
                        <Table.HeaderCell>Registration Date</Tab
                        <Table.HeaderCell>E-mail address</Table.H
                        <Table.HeaderCell>Premium Plan</Table.He

```

## Aprenda a programar — currículo gratuito de 3 mil horas

```
        <Table.Cell>John Lilki</Table.Cell>
        <Table.Cell>September 14, 2013</Table.Ce
        <Table.Cell>jhlilk22@yahoo.com</Table.Ce
        <Table.Cell>No</Table.Cell>
      </Table.Row>
    </Table.Body>
  </Table>
</div>
)
}
```

Read.js



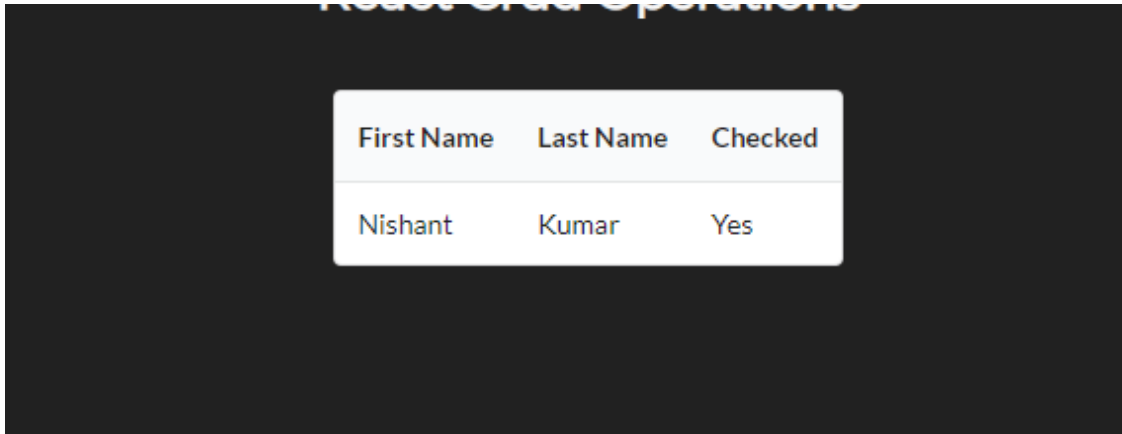
The screenshot shows a web application with a dark background. At the top, the title 'React Crud Operations' is displayed in a light-colored font. Below the title is a table with a light gray header and one data row. The table has four columns: 'Name', 'Registration Date', 'E-mail address', and 'Premium Plan'. The data row contains the following values: 'John Lilki', 'September 14, 2013', 'jhlilk22@yahoo.com', and 'No'.

Name	Registration Date	E-mail address	Premium Plan
John Lilki	September 14, 2013	jhlilk22@yahoo.com	No

Esse é o resultado da página de leitura (Read). Temos uma tabela com quatro colunas, mas somente precisamos de três.

Remova as colunas adicionais de campo e renomeio os campos assim:

Aprenda a programar — currículo gratuito de 3 mil horas



The screenshot shows a dark-themed web application. At the top, there is a header with the text 'React CRUD Operations' in a stylized font. Below the header, there is a table with three columns: 'First Name', 'Last Name', and 'Checked'. The table has one data row with the values 'Nishant', 'Kumar', and 'Yes'.

First Name	Last Name	Checked
Nishant	Kumar	Yes

Esta é a aparência de nossa página de leitura agora:

```
import React from 'react';
import { Table } from 'semantic-ui-react'
export default function Read() {
  return (
    <div>
      <Table singleLine>
        <Table.Header>
          <Table.Row>
            <Table.HeaderCell>First Name</Table.HeaderCell>
            <Table.HeaderCell>Last Name</Table.HeaderCell>
            <Table.HeaderCell>Checked</Table.HeaderCell>
          </Table.Row>
        </Table.Header>

        <Table.Body>
          <Table.Row>
            <Table.Cell>Nishant</Table.Cell>
            <Table.Cell>Kumar</Table.Cell>
            <Table.Cell>Yes</Table.Cell>
          </Table.Row>
        </Table.Body>
      </Table>
    </div>
  )
}
```

Aprenda a programar — currículo gratuito de 3 mil horas

Em seguida, vamos enviar uma solicitação de GET para obter os dados da API.

Precisamos dos dados quando nossa aplicação carregar. Assim, vamos usar o hook `useEffect`.

```
import React, { useEffect } from 'react';

useEffect(() => {

}, [])
```

Hook `useEffect`

Crie um *state* que conterá os dados recebidos. Este state será um array.

```
import React, { useEffect, useState } from 'react';

const [APIData, setAPIData] = useState([]);
useEffect(() => {

}, [])
```

State de `APIData` para armazenar os dados recebidos da API

No hook `useEffect`, vamos enviar a solicitação de GET.



## Aprenda a programar — currículo gratuito de 3 mil horas

```
    })  
  }, [])
```

Assim, usaremos `axios.get` para enviar a solicitação de GET à API. Então, se a solicitação for preenchida, definiremos os dados da resposta em nosso *state* da *APIData*.

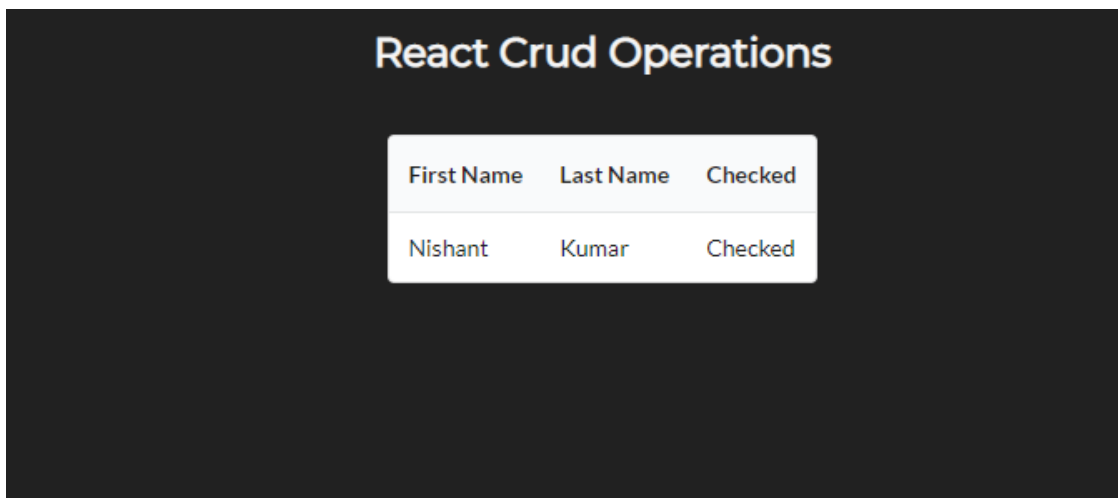
Agora, vamos mapear nossas linhas da tabela de acordo com os dados da API.

Vamos usar a função `Map` para fazer isso. A função vai percorrer o array e exibir os dados no resultados.

```
<Table.Body>  
  {APIData.map((data) => {  
    return (  
      <Table.Row>  
        <Table.Cell>{data.firstName}</Table.Cell>  
        <Table.Cell>{data.lastName}</Table.Cell>  
        <Table.Cell>{data.checkbox ? 'Checked' : 'Unchecked'}</Table.Cell>  
      </Table.Row>  
    )  
  })  
</Table.Body>
```

Estamos mapeando *firstName*, *lastName* e a caixa de seleção de acordo com os dados na API. Nossa caixa de seleção, no entanto, é um pouco diferente. Usei aqui um operador ternário ('?'). Se *data.checkbox* for *true*, o resultado será *Checked* (marcado). Do contrário, será *Unchecked* (desmarcado).

Aprenda a programar — currículo gratuito de 3 mil horas



The screenshot shows a web application with a dark background. At the top, the title 'React Crud Operations' is displayed in a light-colored font. Below the title, there is a table with three columns: 'First Name', 'Last Name', and 'Checked'. The table contains one row of data with the values 'Nishant', 'Kumar', and 'Checked' respectively.

First Name	Last Name	Checked
Nishant	Kumar	Checked

Resultado de Read.js

## A operação de atualização

Crie mais um cabeçalho para a atualização (Update) e uma coluna para cada linha da tabela para um botão de atualização. Use o botão de Semantic UI do React.

```
<Table.HeaderCell>Update</Table.HeaderCell>

<Table.Cell>
  <Button>Update</Button>
</Table.Cell>
```

Criação do botão Update

Agora, ao clicarmos nesse botão, deveremos ser redirecionados para a página de atualização. Para isso, precisamos do Link de React Router.

## Aprenda a programar — currículo gratuito de 3 mil horas

```
import { Link } from 'react-router-dom';

<Link to='/update'>
  <Table.Cell>
    <Button>Update</Button>
  </Table.Cell>
</Link>
```

Link para o botão Update

Desse modo, se clicarmos no botão Update, seremos redirecionados para a página de atualização.

Para atualizar os dados da coluna, precisaremos dos ID respectivos, os quais obteremos na API.

Crie uma função chamada `setData`. Vincule essa função ao botão Update.

```
<Button onClick={() => setData()}>Update</Button>
```

Depois disso, precisamos passar os dados como parâmetro para a função superior.

```
<Button onClick={() => setData(data)}>Update</Button>
```

Na função acima, registramos esses dados no console:

Aprenda a programar — currículo gratuito de 3 mil horas

}

```
▼ {id: "1", firstName: "Nishant", lastName: "Kumar", checkbox: true} read.js:17
  checkbox: true
  firstName: "Nishant"
  id: "1"
  lastName: "Kumar"
  ▶ __proto__: Object
```

Dados no console

Clique no botão Update na tabela e confira o console. Você obterá os dados do campo da tabela respectivo.

Vamos definir esses dados no localStorage.

```
const setData = (data) => {
  let { id, firstName, lastName, checkbox } = data;
  localStorage.setItem('ID', id);
  localStorage.setItem('First Name', firstName);
  localStorage.setItem('Last Name', lastName);
  localStorage.setItem('Checkbox Value', checkbox)
}
```

Definindo os dados no localStorage (Armazenamento local)

Estamos fazendo a desestruturação dos nossos dados em *id*, *firstName*, *lastName*, e *checkbox*. Em seguida, definimos esses dados no armazenamento local. Você pode usar o armazenamento local para armazenar os dados localmente no navegador.

## Aprenda a programar — currículo gratuito de 3 mil horas

## Update.

```
import React, { useState } from 'react';
import { Button, Checkbox, Form } from 'semantic-ui-react'
import axios from 'axios';

export default function Update() {
  const [firstName, setFirstName] = useState('');
  const [lastName, setLastName] = useState('');
  const [checkbox, setCheckbox] = useState(false);

  return (
    <div>
      <Form className="create-form">
        <Form.Field>
          <label>First Name</label>
          <input placeholder='First Name' onChange={(e)
        </Form.Field>
        <Form.Field>
          <label>Last Name</label>
          <input placeholder='Last Name' onChange={(e)
        </Form.Field>
        <Form.Field>
          <Checkbox label='I agree to the Terms and Cor
        </Form.Field>
        <Button type='submit'>Update</Button>
      </Form>
    </div>
  )
}
```

Nossa página de atualização

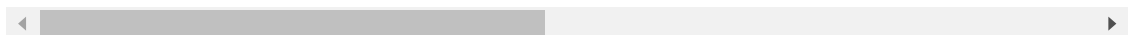
Crie um hook `useEffect` no componente `Update`. Nós o utilizaremos para obter os dados que armazenamos anteriormente em `localStorage`. Além disso, crie mais um *state* para o campo *ID*.

### Aprenda a programar — currículo gratuito de 3 mil horas

```
setID(localStorage.getItem('ID'))
setFirstName(localStorage.getItem('First Name'));
setLastName(localStorage.getItem('Last Name'));
setCheckbox(localStorage.getItem('Checkbox Value'))
}, []);
```

Defina os dados respectivos de acordo com suas chaves no armazenamento local. Precisamos definir esses valores nos campos do formulário. Isso preencherá os campos automaticamente quando a página de Update for carregada.

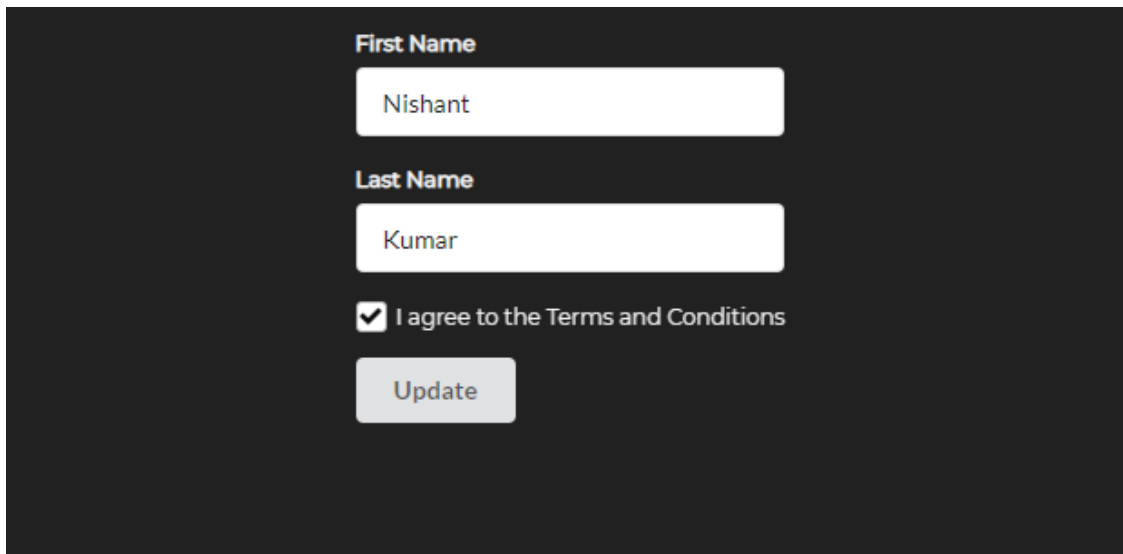
```
<Form className="create-form">
  <Form.Field>
    <label>First Name</label>
    <input placeholder='First Name' value={firstN
  </Form.Field>
  <Form.Field>
    <label>Last Name</label>
    <input placeholder='Last Name' value={lastNar
  </Form.Field>
  <Form.Field>
    <Checkbox label='I agree to the Terms and Cor
  </Form.Field>
  <Button type='submit'>Update</Button>
</Form>
```



Definindo os valores nos campos do formulário

Agora, se clicarmos no botão Update na página de leitura, seremos redirecionados para a página de atualização, onde veremos todos os dados do formulário preenchidos automaticamente.

## Aprenda a programar — currículo gratuito de 3 mil horas

A imagem mostra uma interface de usuário para atualizar dados. Há dois campos de texto: 'First Name' com o valor 'Nishant' e 'Last Name' com o valor 'Kumar'. Abaixo deles, há uma caixa de seleção marcada com o texto 'I agree to the Terms and Conditions'. No final, há um botão cinza com o texto 'Update'.

Página de atualização

Em seguida, criaremos a solicitação para a atualização dos dados.

Crie uma função chamada `updateAPIData`. Dentro dessa função, usaremos o `axios.put` para enviar uma solicitação de PUT, a qual atualizará nossos dados.

```
const updateAPIData = () => {  
  axios.put(`https://60fbca4591156a0017b4c8a7.mockapi.io/fakeData/  
    firstName,  
    lastName,  
    checkbox  
  `)  
}
```

Aqui, você pode ver que estamos adicionando ao endpoint da API um campo `id`.

## Aprenda a programar — currículo gratuito de 3 mil horas

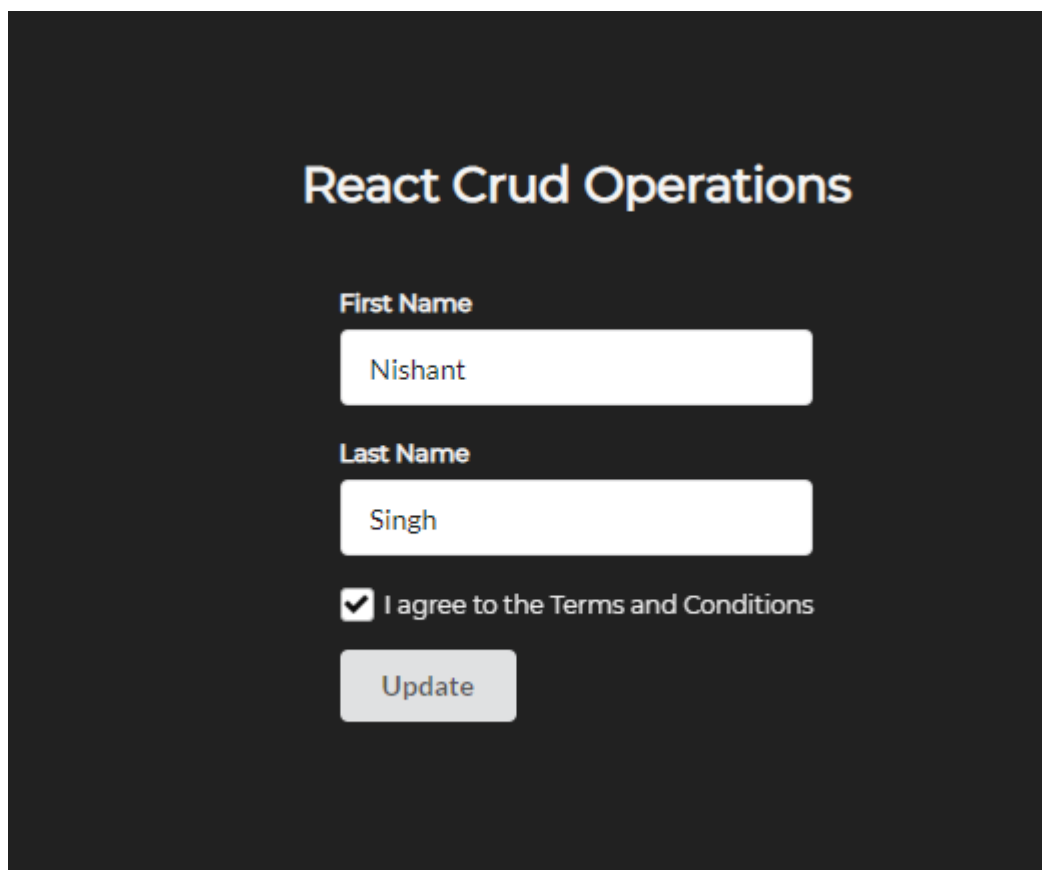
Depois disso, passamos o id para o endpoint. Isso nos permite atualizar o campo cujo ID nós acabamos de passar.

Vincule a função `updateAPIData` ao botão Update.

```
<Button type='submit' onClick={updateAPIData}>Update</Button>
```

Vinculando `updateAPIData` ao botão Update

Clique no botão Update na tabela na página de leitura, altere seu sobrenome e clique no botão Update na página de atualização.



The screenshot shows a dark-themed web form titled "React Crud Operations". It contains two text input fields: "First Name" with the value "Nishant" and "Last Name" with the value "Singh". Below these fields is a checkbox labeled "I agree to the Terms and Conditions" which is checked. At the bottom of the form is a grey button labeled "Update".

Atualização dos campos

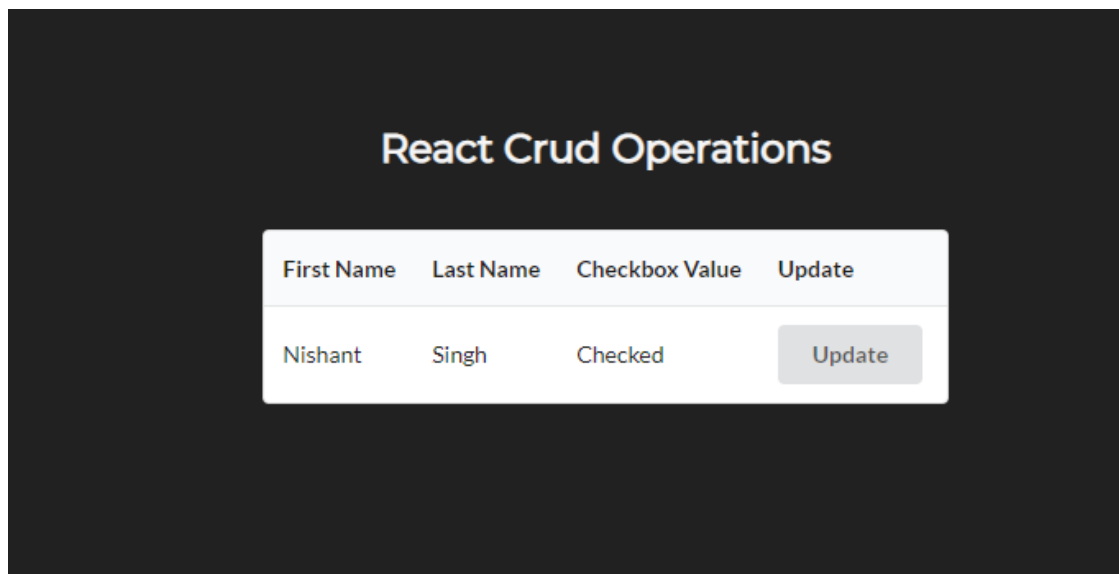


Aprenda a programar — currículo gratuito de 3 mil horas

React, Hooks, Axios e React Router

```
[{"id": "1", "firstName": "Nishant", "lastName": "Singh", "checkbox": true}]
```

Na Mock API



Em nossa página de leitura

## A operação de exclusão

Adicione mais um botão na tabela em Read, que usaremos para a operação de exclusão.

```
<Table.Cell>  
  <Button onClick={() => onDelete(data.id)}>Delete</Button>  
</Table.Cell>
```

Botão de exclusão na tabela do componente Read

## Aprenda a programar — currículo gratuito de 3 mil horas

```
const onDelete = (id) => {  
  
}
```

A função de exclusão

Usaremos o `axios.delete` para excluir as colunas respectivas.

```
const onDelete = (id) => {  
  axios.delete(`https://60fbca4591156a0017b4c8a7.mockapi.io/fakeI  
}
```



Exclusão dos campos da API

Clique no botão Delete e confira a API. Você verá que os dados foram excluídos.

Precisamos carregar os dados da tabela após eles terem sido excluídos.

Desse modo, crie uma função para carregar os dados da API.

```
const getData = () => {  
  axios.get(`https://60fbca4591156a0017b4c8a7.mockapi.io/fakeD  
    .then((getData) => {  
      setAPIData(getData.data);
```

## Aprenda a programar — currículo gratuito de 3 mil horas

## Obtendo os dados da API

Agora, na função `onDelete`, precisamos carregar os dados atualizados após excluirmos um campo.

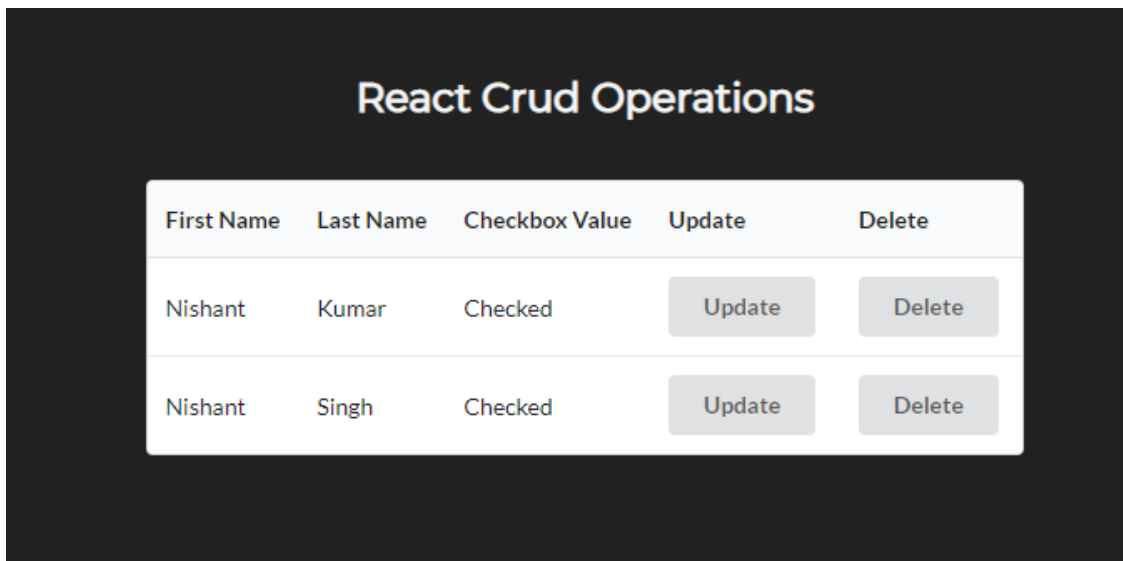
```
const onDelete = (id) => {  
  axios.delete(`https://60fbca4591156a0017b4c8a7.mockapi.io/`)  
    .then(() => {  
      getData();  
    })  
}
```

Carregando os dados atualizados após excluir um campo

React Crud Operations				
First Name	Last Name	Checkbox Value	Update	Delete
Nishant	Kumar	Unchecked	<button>Update</button>	<button>Delete</button>
Nishant	Kumar	Checked	<button>Update</button>	<button>Delete</button>
Nishant	Singh	Checked	<button>Update</button>	<button>Delete</button>

Tabela do componente Read

Aprenda a programar — currículo gratuito de 3 mil horas



The screenshot shows a web application titled "React Crud Operations" with a dark background. In the center, there is a white table with the following structure:

First Name	Last Name	Checkbox Value	Update	Delete
Nishant	Kumar	Checked	<button>Update</button>	<button>Delete</button>
Nishant	Singh	Checked	<button>Update</button>	<button>Delete</button>

Tabela do componente Read após excluir um campo

## Vamos fazer algumas melhorias ao nosso app com CRUD

Então, ao publicarmos nossos dados na página Create, simplesmente estaremos obtendo os dados de nosso banco de dados falso. Precisamos redirecionar para a página de leitura quando nossos dados são criados na página de criação.

Importe `useHistory` de React Router.

```
import { useHistory } from 'react-router';
```

Importando `useHistory` de React Router

## Aprenda a programar — currículo gratuito de 3 mil horas

```
let history = useHistory();
```

Em seguida, use a função `history.push` para fazer o push para a página de leitura logo após a API de publicação ser chamada.

```
const postData = () => {  
  axios.post(`https://60fbca4591156a0017b4c8a7.mockapi.io/1  
    firstName,  
    lastName,  
    checkbox  
  }).then(() => {  
    history.push('/read')  
  })  
}
```



Fazendo o push para a página de leitura após o post na API ter sido bem-sucedido

Faremos o push para a página de leitura usando o hook `useHistory`.

Faça o mesmo para a página de atualização.

```
import React, { useState, useEffect } from 'react';  
import { Button, Checkbox, Form } from 'semantic-ui-react';  
import axios from 'axios';  
import { useHistory } from 'react-router';  
  
export default function Update() {  
  let history = useHistory();  
  const [id, setID] = useState(null);  
  const [firstName, setFirstName] = useState('');
```

## Aprenda a programar — currículo gratuito de 3 mil horas

```
setFirstName(localStorage.getItem('First Name'));
setLastName(localStorage.getItem('Last Name'));
setCheckbox(localStorage.getItem('Checkbox Value'));
}, []);

const updateAPIData = () => {
  axios.put(`https://60fbca4591156a0017b4c8a7.mockapi.io/first-name`, {
    firstName,
    lastName,
    checkbox
  }).then(() => {
    history.push('/read')
  })
}
return (
  <div>
    <Form className="create-form">
      <Form.Field>
        <label>First Name</label>
        <input placeholder='First Name' value={firstName}/>
      </Form.Field>
      <Form.Field>
        <label>Last Name</label>
        <input placeholder='Last Name' value={lastName}/>
      </Form.Field>
      <Form.Field>
        <Checkbox label='I agree to the Terms and Conditions' checked={checkbox}/>
      </Form.Field>
      <Button type='submit' onClick={updateAPIData}>Update</Button>
    </Form>
  </div>
)
}
```

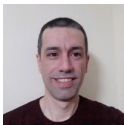
Update.js

Agora, você sabe como realizar operações de CRUD usando o React e os hooks do React!

## Aprenda a programar — currículo gratuito de 3 mil horas

Você pode [encontrar o código deste tutorial no GitHub](#) se quiser fazer mais testes com ele.

*Feliz aprendizagem.*



### Daniel Rosa

Um profissional dos idiomas humanos apaixonado por linguagens de computador. | A world languages professional in love with computer languages.

Se você leu até aqui, agradeça ao autor para mostrar que você se importa com o trabalho. [Agradeça](#)

Aprenda a programar gratuitamente. O plano de estudos em código aberto do freeCodeCamp já ajudou mais de 40.000 pessoas a obter empregos como desenvolvedores. [Comece agora](#)

O freeCodeCamp é uma organização beneficente 501(c)(3), isenta de impostos e apoiada por doações (Número de identificação fiscal federal dos Estados Unidos: 82-0779546).

Nossa missão: ajudar as pessoas a aprender a programar de forma gratuita. Conseguimos isso criando milhares de vídeos, artigos e lições de programação interativas, todas disponíveis gratuitamente para o público.

As doações feitas ao freeCodeCamp vão para nossas iniciativas educacionais e ajudam a pagar servidores, serviços e a equipe.

## Aprenda a programar — currículo gratuito de 3 mil horas

Nova aba em HTML	Jogo do dinossauro
Máscaras de sub-rede	Menu iniciar
40 projetos em JavaScript	Arrays vazios em JS
Tutorial de button onClick	Caracteres especiais
Bot do Discord	Python para iniciantes
Centralizar em CSS	Provedores de e-mail
Excluir pastas com o cmd	15 portfólios
Imagens em CSS	Node.js no Ubuntu
25 projetos em Python	10 sites de desafios
Excluir branches	Clonar branches
Date now em JavaScript	Media queries do CSS
Var, let e const em JavaScript	Fix do Live Server no VS Code
Axios em React	SQL em Python
ForEach em JavaScript	Interpretadas x compiladas
Fotos do Instagram	Imagens SVG em HTML e CSS

### Nossa instituição

[Sobre](#) [Rede de ex-alunos](#) [Código aberto](#) [Loja](#) [Apoio](#) [Patrocinadores](#)

[Honestidade acadêmica](#) [Código de conduta](#) [Política de privacidade](#) [Termos de serviço](#)

[Política de direitos de autor](#)